Software Quality and Infrastructure Protection for Diffuse Computing

J. Feigenbaum, J.Y. Halpern, P.D. Lincoln, J.C. Mitchell, A. Scedrov[*], J.M. Smith[†], and P. Syverson[‡]

Abstract

Diffuse computing is concerned with managing and maintaining a computational infrastucture that is distributed among many heterogeneous nodes that do not trust each other completely and may have differing incentives, needs, and priorities. As commercial, academic, civilian, government, and military systems become increasingly diffuse, the challenges of providing reliable and trustworthy diffuse systems become increasingly important.

Diffuse computational systems require new software design and engineering methodologies. Components are combined on an as-needed basis, and with the increasing scale and complexity of modern distributed systems such as the WWW, the rigor and methodological support have failed to materialize in any form. With the local autonomy that characterizes large-scale distributed sytems, global design and analysis may in fact be impossible.

In this paper we survey the research partially supported by OSD/ONR CIP/SW URI "Software Quality and Infrastructure Protection for Diffuse Computing" through ONR Grant N00014-01-1-0795. We develop fundamental understanding, models, algorithms, and network testbed, in order to reduce cost, improve performance, and provide higher reliability for networked operations across untrusted networks. We take a synergistic approach, combining several complementary thrusts: incentive-compatibility in distributed computation, authorization mechanisms, infrastructure and secure communication protocols, privacy and anonymity, and networking. We study a market system of autonomous agents that realistically model the behavior of a large-scale distributed system. Rules imposed on the market system ("mechanism design") allow global desiderata to be achieved in spite of local autonomy. In this way, the behavior of the software as a system can be described formally in spite of incomplete knowledge. We carry out the initial development of such a methodology and the construction of a multi-institutional experimental platform upon which we can prototype this software-quality methodology. This will open up a whole new range of "global" software-design techniques that work in today's and tomorrow's systems.

These advances, leading to new software technology, will ensure greater confidence in critical national infrastructures that depend so much on information technology.

[*]P.I. of the project. University of Pennsylvania, Philadelphia, PA 19104. scedrov@math.upenn.edu

[†]The work reported by J.M. Smith was completed before December 31, 2003.

[‡]External collaborator.

1 Diffuse computing

Advances in networking, web services, business infrastructure, and mass market demands have led to the emergence of a new computing paradigm that we will refer to as *diffuse computing.* In this paradigm, the focus is shifting from self-contained computers and client-server clusters to constellations of services that work together. Already, private companies provide free email management and storage, free mailing list and web page management, free storage for digital photo collections, and other web-based services that obviate the need for individuals to install, maintain and host software to provide these functions. On a larger, corporate scale, more and more computing services are being outsourced. Web-site hosting, caching, network delivery, and other services are provided by contract organizations that manage and maintain the required software, hardware, and network connectivity. On a more dynamic level, mobile code mechanisms and dynamic service platforms such as Jini allow the installed code of a running system to change rapidly, in the process of performing a single or multiple simultaneous tasks. Driven by the potential for improved functionality, better interoperability, and the need for networked operations in all walks of life, the common "computer system" used by an individual or organization no longer consists of an identifiable set of applications running on an identifiable hardware configuration linked to the network through a single point of access or firewall. Instead, daily computing draws on a constellation of services provided by various sites on the network, each autonomously administered and updated by the provider of the service.

A related phenomenon is the rapid rise in prominence of peer-to-peer software systems, in which a single task may be distributed among a dynamically changing set of available computers, linked through network paths of different capacity. Prominent and emerging peer-to-peer examples include *distributed.net*, http://www.distributed.net/index.html.en, an organization allowing thousands of computers around the world to participate in mathematics and cryptography challenges, the *Folding@Home* project, please see information available on http://www.stanford.edu/group/pandegroup/Cosm/, distributing a "screen saver" that allows users to contribute computing resources to protein-folding simulations, and the *Casino-21* project http://www.climate-dynamics.rl.ac.uk/~hansen/casino21.html in which more than 20,000 people worldwide already have volunteered to contribute their personal computers' off-hours power to a scientific experiment that will attempt to forecast the climate of the 21st century.

Staggering computational power and information resources can be assembled from a collection of small to moderate sized computing devices, coordinated over a network: distributed.net's computing power is more than 160,000 PCs working 24 hours a day, 7 days a week, 365 days a year! From a positive standpoint, pressing world problems can conceivably be solved by peer-to-peer systems. On a smaller scale, a collection of networked personal devices, carried on the body and/or installed in a vehicle, can work together to provide significant coordinated computing power as needed, making the whole greater than the sum of its parts, not only in power but in reliability and resistance to faults or corruption.

Diffuse computing has other advantages besides computational power. It also provides mobility, scalability, ease of maintenance, and fault tolerance. Mobility follows from the diffuse nature of the medium. If an individual draws computing services from a network, then those services may be preserved as the individual (or computing device, or an embedded computer) moves around physically or around the network. Scalability arises from the ability to switch vendors or aggregate services. For example, if information storage and retrieval is outsourced to a storage vendor, then a client may buy more storage if more is needed, or aggregate the services of more than one vendor if greater reliability through replication is desired. Maintenance becomes the obligation of service providers, who may accrue resulting economies of scale. Dynamic service installation also simplifies maintenance in a way that usefully decouples subsystems: if a client caches an interface implemented in mobile code, and refreshes periodically, this automatic process keeps the client current with upgrades, bug fixes, and performance enhancements provided by the service. By us-

ing homogeneous peer-to-peer designs without a centralized server, systems no longer suffer from a single point of failure.

Given these advantages, it seems clear that diffuse computing is here to stay. In time, we expect individuals and organizations to be able to draw all of their computing needs from a diffuse collection of network-available services, including information aggregation, information storage and retrieval, and high-powered compute services that provide compute cycles on a contract basis. Diffuse computing not only has an appeal for individuals and private enterprises who may draw on a collection of internet services, but also provides a useful computational model for a large organization that is heterogeneous and geographically distributed.

Unfortunately, along with all of its benefits, diffuse computing also opens the door to enormous risks. The same principle that allows difficult number-theoretic problems to be solved by distributed.net also allows a similar coalition to flood parts of the network with massive denial-of-service attacks, crack codes of banks or government installations, process surveillance information without the knowledge of innocent individuals running innocuous screen savers, and so on. The February 2000 "tribal flood network" distributed denial-of-service attack using a coordinated network of compromised machines illustrates the potential for network-based attack. In addition, diffuse computing heightens security concerns, since an individual or operational unit cannot trust the providers of some or all of the services they rely upon.

In short, diffuse computing offers great potential to those who will understand its strengths and pragmatic operational properties, and great risk to those who fail to understand the potential threats and conceivable countermeasures.

We undertake a comprehensive study of the software infrastructure needed for diffuse computing. We combine several complementary thrusts: incentive-compatibility in distributed computation, authorization mechanisms, infrastructure and communication protocols, privacy and anonymity, and networking. One particular focus is the study of market systems of autonomous agents that realistically model the behavior of large-scale distributed systems. Rules imposed on the market system ("mechanism design") allow global desiderata to be achieved in spite of local autonomy. We carry out the initial development of such a software-quality methodology, including a systematic, formal treatment of underlying models, algorithms, and data structures, and the construction of a multi-institutional experimental platform upon which we can prototype it. This will open up a whole new range of "global" software-design techniques that work in today's and tomorrow's systems.

These advances, leading to new software technology, will ensure greater confidence in critical national infrastructures that depend so much on information technology.

2 Challenges in diffuse software infrastructure

In this section, we consider in more detail some of the issues involved in designing appropriate software for diffuse systems and our work. The subsequent sections provide more details of our approach.

2.1 Market-based computation

In designing efficient, distributed algorithms and network protocols, computer scientists typically assume that users are either *cooperative* (*i.e.*, that they follow the protocol) or that they are faulty. The fault-tolerance community has considered various fault models, ranging from *crash* failures (where a faulty process simply crashes and does not take part in the protocol thereafter) to *Byzantine* failures (where a faulty process may actively try to disrupt a protocol) [82]. In contrast, game theorists design market mechanisms in which it is

assumed that users are neither cooperative nor adversarial but rather *selfish*: They respond to well-defined incentives and will deviate from the protocol only for tangible gain. Until recently, computer scientists ignored incentive-compatibility, and game theorists ignored computational efficiency. Recent work on *algorithmic mechanism design* [151, 78] combines computational complexity and incentive-compatibility. We consider the combination of various fault and attacker models with incentive compatibility. That is, we consider resource-restricted selfish attacker behavior within a fault model containing simple as well as Byzantine faults. We use game-theoretic principles in designing efficient algorithms for the allocation of scarce resources in a diffuse computing environment. In addition, we explore the completely open question of whether user-privacy concerns can be "layered" on top of incentive-compatibility or whether privacy has to be combined with other desiderata in the original mechanism design.

2.2 Authorization management

In centralized computing systems, and even in distributed systems that are closed or relatively small, authorization to use resources (*e.g.*, databases, web pages, corporate network gateways, *etc.*) is broken down into *authentication* ("who made this request?") and *access control* ("is the requestor authorized to use the resource?"). An emerging body of work on *trust management* (see, *e.g.*, [166, 73, 34, 121]) recognizes that this traditional approach will not work in the type of diffuse computing environment that we address in this proposal. We adopt the trust-management approach, in which a request may be accompanied by one or more "credentials," and the authorization question becomes "do these credentials prove that this request complies with the policy governing this resource?". In particular, we investigate the open or partially open issues of nonmonotonic policies, tools for formal analysis of security policies, and credential-storage and credential-retrieval.

2.3 Protocols for diffuse computation and secure communication

Diffuse computation requires new network protocols for locating and for maintaining an inventory of diffuse services and for secure and reliable communication among components of a diffuse system. Relevant research topics include:

1. *Developing efficient service-discovery protocols,* which allow components of a diffuse system to maintain contact and efficient routing among a dynamically changing set of cooperating components,
2. *Developing efficient service search techniques* that allow one component to find data or computational services from cooperating components,
3. *Developing secure and authenticated communication protocols* for diffuse services, including integration methods for applying distributed authorization management concepts,
4. *Extending and applying protocol design and analysis methods* to diffuse service protocols and related network protocols such as group key management and secure multicast protocols, and
5. *Developing new specification techniques* appropriate for specifying diffuse protocols, that take into account costs and uncertainty.

Reliable, secure communication is a cornerstone of assured diffuse computation since distributed services must be able to communicate reliably in the face of network unreliability, network-based interference or attack.

2.4 Privacy and anonymity

Exponential growth in digital information gathering, storage, and processing capabilities inexorably leads to conflict between well-intentioned government or commercial datamining, and fundamental privacy interests of individuals and organizations. We develop a mechanism that provides cryptographic fetters on the mining of personal data, enabling efficient mining of previously-negotiated properties, but preventing any other uses of the protected personal data. We also provide a framework for reasoning about information-hiding requirements in multiagent systems and for reasoning about anonymity in particular. Furthermore, we explore some reasons why diffuse anonymity systems are particularly hard to deploy, enumerate the incentives to participate either as senders or also as nodes, and build a general model to describe the effects of these incentives. We introduce a new cryptographic technique which leads to new types of functionality in mixnet architectures. We also investigate anonymity network topologies.

2.5 Networking

Increases in scale, complexity, dependency and security for networks have motivated increased automation of activities. Technology derived from active networking research can be used to develop a series of *network monitoring* systems. This approach allows users to customize the monitoring function at the lowest possible level of abstraction to suit a wide range of monitoring needs: we use operating-system mechanisms that result in a programming environment providing a high degree of flexibility, retaining fine-grained control over security, and minimizing the associated performance overheads. We also investigate *proxy-based transcoding*, which adapts Web content to be a better match for client capabilities (such as screen size and color depth) and last-hop bandwidths. Traditional transcoding breaks the end-to-end model of the Web, because the proxy does not know the semantics of the content. *Server-directed transcoding* preserves end-to-end semantics while supporting aggressive content transformations. We also develop techniques for measuring Internet path properties, and to use these techniques to study network performance and protocol behavior. Our current focus in this regard is on *network tomography* tools that allow us to measure network-internal delays.

2.6 Outline of the paper

In the following sections, we describe in more detail certain aspects of our work that addresses the challenges discussed just above. Our work on market-based computation is presented in section 3. In Section 4 we discuss our work on authorization management. Our research on protocols is discussed in Section 5. Our work on privacy and anonymity is discussed in Section 6. Our networking research is discussed in Section 7.

The full list of publications stemming from our work on this project from its inception in May 2001 to date includes more than 60 publications and 5 software prototypes, all available on the project web site `http://www.cis.upenn.edu/spyce/`. In this survey we feature only a selection of our results.

We would like to thank Dr. Ralph Wachter of the Office of Naval Research for his most valuable advice and guidance. We would also like to thank Cynthia Dwork, Tim Griffin, and Vitaly Shmatikov for inspiring and productive scientific discussions and for their contribution to the project.

3 Market-based computation

The emergence of the Internet as a standard, widely used distributed computing environment and of Internet-enabled commerce (both in traditional, "real-world" goods and in electronic goods and computing services themselves) has drawn computer scientists' attention to incentive-compatibility questions in distributed computation. In particular, there are (largely independent) growing bodies of relevant literature on incentive-compatibility questions in distributed computation, in the theoretical computer science community [151, 78] and in the "distributed agents" part of the AI community [155, 144, 186]. In the information-assurance community, Meadows is developing methods to make cryptographic protocols more resistant to denial of service by trading off the cost to defender against the cost to the attacker [133]. Managing incentives also influences the design of the support system for massive multiplayer games discussed in Section 6. Incentive-compatibility in traffic analysis resistant communication will also be discussed in Section 6.

In the first paper on what they aptly call *algorithmic mechanism design,* Nisan and Ronen [151] add computational efficiency to the set of concerns that must be addressed in the study of how privately known preferences of a large group of people can be aggregated into a "social choice." (This is the subfield of microeconomics known as *mechanism design* or *implementation theory*; see *e.g.*, [128] or [152, Chap. 10].) Using the results in [151] on task allocation as a starting point, we explore a number open questions related to incentive compatibility, including a mechanism for lowest-cost routing based on the Border Gateway Protocol (BGP), computational complexity of mechanism design for policy routing, and autonomous nodes and distributed mechanisms.

3.1 A BGP-based mechanism for lowest-cost routing

The results discussed in this subsection are due to Feigenbaum, Papadimitriou, Sami, and Shenker [77].

The Internet is comprised of many separate administrative domains known as *Autonomous Systems* (ASs). Routing occurs on two levels, intradomain and interdomain, implemented by two different sets of protocols. Intradomain-routing protocols, such as OSPF, route packets within a single AS. Interdomain routing, currently handled by the Border Gateway Protocol (BGP), routes packets between ASs. Although routing is a very well-studied problem, it has been approached by computer scientists primarily from an engineering or "protocol-design" perspective.

In their seminal paper on algorithmic mechanism design, Nisan and Ronen [151] advocate combining an economic, "incentive-compatibility" approach with the more traditional protocol-design approach to the problem. Internet routing is an extremely natural problem in which to consider incentives, because ownership, operation, and use by numerous independent, self-interested parties give the Internet the characteristics of an economy as well as those of a computer. We continue the study of routing from a mechanism-design perspective, concentrating specifically on interdomain routing, for reasons explained below.

In our formulation of the routing-mechanism design problem, each AS incurs a per-packet *cost* for carrying traffic, where the cost represents the additional load imposed on the internal AS network by this traffic. To compensate for these incurred costs, each AS is paid a *price* for carrying *transit* traffic, which is traffic neither originating from nor destined for that AS. It is through these costs and prices that consideration of "incentive compatibility" is introduced to the interdomain-routing framework, which, as currently implemented, does not explicitly consider incentives. Our goal is to maximize network efficiency by routing packets along the lowest-cost paths (LCPs). We are following previous work on mechanism design for routing [151, 105] by introducing incentives in this way, and focusing on lowest-cost paths.

The main difference between this work and previous algorithmic mechanism design for routing is

our focus on computing the routes and prices in a *distributed* fashion. Furthermore, in order to steer the mechanism-design approach towards practical implementation, we consider only distributed algorithms that retain the data structures and communication protocol of BGP. We develop a "BGP-based computational model" to capture these requirements and seek to compute the routes and prices in this model.

Given a set of costs, the LCPs can be computed using standard routing protocols (such as BGP). However, under many pricing schemes, an AS could be better off lying about its costs;[1] such lying would cause traffic to take non-optimal routes and thereby interfere with overall network efficiency.

To prevent this, we first ask how one can set the prices so that ASs have no incentive to lie about their costs; such pricing schemes are called *strategyproof*. We also require that ASs that carry no transit traffic receive no payment. We prove that there is only one strategyproof pricing scheme with this property; it is a member of the Vickrey-Clarke-Groves (VCG) class of mechanisms. This mechanism requires a per-packet price to be paid to each transit node k; this price is determined by the cost of the LCP and the cost of the lowest-cost path that does not pass through k. We next ask how the VCG prices should be computed, and we provide a "BGP-based" distributed algorithm that accomplishes this.

Our results contribute in several ways to the understanding of how incentives and computation affect each other in routing-protocol design. Nisan and Ronen [151] and Hershberger and Suri [105] considered the LCP mechanism-design problem, motivated in part by the desire to include incentive issues in Internet-route selection. The LCP mechanism studied in [151, 105] takes as input a biconnected graph, a single source, a single destination, and a (claimed) transmission cost for each link; the strategic agents are the links, and the mechanism computes, in a strategyproof manner, both an LCP for this single routing instance and a set of payments to the links on the LCP. This mechanism is a member of the VCG family and forms the point of departure for our work. However, our formulation of the problem differs in three respects, each of which makes the problem more representative of real-world routing:

1. First, in our formulation, it is the nodes that are the strategic agents, not the links as in [151, 105]. We make this choice because we are trying to model *interdomain* routing. ASs actually *are* independent economic actors who could strategize for financial advantage in interdomain-routing decisions; in the BGP computational model into which we seek to incorporate incentive issues, it is the nodes that represent ASs and that are called upon to "advertise" their inputs to the protocol. Formulations in which the links are the strategic agents might be more appropriate for intradomain routing, but it is not clear that incentive issues are relevant in that context; because all links and routers within a domain are owned and managed by a single entity, they are unlikely to display strategic behavior.

2. Second, instead of taking as input a single source-destination pair and giving as output a single LCP, our mechanism takes in n AS numbers and constructs LCPs for all source-destination pairs. Once again, we make this choice in order to model more accurately the interdomain-routing scenario. This complicates the problem, because there are now n^2 LCP instances to solve.

3. Third, we compute the routes and the payments not with a centralized algorithm, as is done in [151, 105], but with a distributed protocol based on BGP. This is necessary if the motivation for the mechanism-design problem is Internet routing, because interdomain-route computation is in fact done in a distributed fashion, with the input data (AS-graph topology) and the outputs (interdomain routes) stored in a distributed fashion as well. The various domains are administratively separate and in some cases competitors, and there is no obvious candidate for a centralized, trusted party that could

[1] There are two ways in which lying might increase the AS's total welfare: Announcing a lower-than-truthful cost might attract more than enough additional traffic to offset the lower price, or announcing a higher-than-truthful cost might produce an increase in the price that is more than sufficient to offset any resulting decrease in traffic.

maintain an authoritative AS graph and tell each of the ASs which routes to use. Real-world BGP implementations could be extended easily to include our pricing mechanism, and we prove that such an extension would cause only modest increases in routing-table size and convergence time.

Our approach of using an existing network protocol as a substrate for realistic distributed computations may prove useful generally in Internet-algorithm design, not only in routing or pricing problems. Algorithm design for the Internet has the extra subtlety that adoption is not a decision by a systems manager, concerned only with performance and efficiency, but rather a careful compromise by a web of autonomous entities, each with its own interests and legacies. Backward compatibility with an established protocol is a constraint and criterion that is likely to become increasingly important and prevalent.

Despite these efforts to formulate the problem realistically, there are several aspects of reality that we deliberately ignore. First, per-packet costs are undoubtedly not the best cost model, *e.g.*, in some cases transit costs are more administrative than traffic-induced. Second, BGP allows an AS to choose routes according to any one of a wide variety of local policies; LCP routing is just one example of a valid policy, and, in practice, many ASs do not use it [181]. Furthermore, most ASs do not allow non-customer transit traffic on their network.[2] Here we ignore general policy routing and transit restrictions; we only use LCPs. Lastly, BGP does not currently consider general path costs; in the cases in which AS policy seeks LCPs, the current BGP simply computes *shortest* AS paths in terms of number of AS hops. This last aspect is minor, because it would be trivial to modify BGP so that it computes LCPs; in what follows, we assume that this modification has been made.

Because of these limitations, our results clearly do not constitute a definitive solution to the incentive problem in interdomain routing. Nonetheless, they represent measurable progress on two fronts. First, although it does not capture all of the important features of interdomain routing, our problem formulation is an improvement over the previous ones in the algorithmic mechanism-design literature [151, 105], as explained above. Second, we have expanded the scope of *distributed algorithmic mechanism design*, which has heretofore been focused mainly on multicast cost sharing [78, 17, 76].

3.2 Mechanism design for policy routing

The results discussed in this subsection appear are due to Feigenbaum, Sami, and Shenker [79].

In doing interdomain routing using BGP, one of the key decisions an AS must make is how to select a route from all the routes it knows of to a particular destination. One frequently studied model has each AS look at some objective metric over the routes, such as the number of ASs a route passes through or the cost of a route, and pick the route that minimizes this metric. In practice, however, ASs want to select a route based on many other criteria, such as commercial relationships or perceived reliability. For example, it is common for an AS to select a route advertised by one of its customers over all other routes. Thus, BGP was explicitly designed to allow ASs to apply their own *routing policies* to the route-selection and route-advertisement processes. This feature of interdomain routing is referred to as *policy-based routing* or *policy routing* for short.

Another aspect of routing that has recently received attention is that of incentives. The participants in the routing process—the ASs, in this case—are independent economic entities, each with its own goals. Thus, they cannot be relied on to follow any specified policy in circumstances in which they could profit by deviating from that policy. Further, much of the information relevant to selecting good routes, such as costs or connectivity information, is known privately to individual ASs; thus, even if there were a central authority

[2] We say that two ASs are "interconnected" if there is a traffic-carrying link between them. Interconnected ASs can be *peers*, or one can be a customer of the other. Most ASs do not accept transit traffic from peers, only from customers.

capable of enforcing a policy, it could not possibly detect strategic reporting of this information. We explore the extent to which one can cope with these strategic issues in a computationally feasible manner.

Algorithmic mechanism design seeks to address both incentives and computational complexity. One of the problems studied by Nisan and Ronen [151] is a simple routing problem: Given a graph with a distinguished source node s, a distinguished sink node t, and costs associated with each edge, find the lowest-cost path from s to t. The wrinkle in the model is that each edge can strategically lie about its cost. Nisan and Ronen showed how a central authority can compute payments for each edge such that every edge's dominant strategy is to be honest about its cost, yielding a *strategyproof mechanism* for this problem. Later, Hershberger and Suri [105] presented a more efficient algorithm to compute the payments required by this mechanism. Archer and Tardos [18] and Elkind *et al.* [72] study mechanisms to select a path that minimizes a metric from a broad class, not necessarily the sum of edge costs; this too can be viewed as a variant of lowest-cost routing.

As discussed in Section 3.1, this approach was extended by Feigenbaum, Papadimitriou, Sami, and Shenker [77], who give a strategyproof mechanism for the lowest-cost routing problem that can be computed by an efficient distributed algorithm. Moreover, they show that this mechanism can be computed by a "BGP-based" algorithm, *i.e.*, an algorithm with similar data structures and communication patterns to BGP that requires only modest increases in communication and convergence time. Thus, the mechanism is "backward compatible" with BGP, which is critical for any routing algorithm that must be implemented in the current Internet.

All the work on mechanism design for routing has focused on variants of *lowest-cost* routing. In practice, this has two drawbacks: The cost model is oversimplified, and the requirement that all ASs use a lowest-cost routing policy is too restrictive. We investigate whether the distributed algorithmic mechanism design approach can be extended to general policy routing. In essence, we look at interdomain routing at a higher level of abstraction: We assume that source ASs have preferences over alternative routes to a destination, but we do not model the *causes* of these preferences. Thus, in our initial model, an AS can express any routing policy, provided that it is based on *some* underlying utility function—it need not arise from the cost of the route but may take into account unspecified, subjective route attributes as well. The goal of the mechanism is to compute routes for every source-destination pair such that the *overall welfare, i.e.*, the sum of all ASs' utility for their selected routes, is maximized. The only constraint on the selected routes is that all routes to a given destination must form a tree; this is a very natural constraint in the Internet, where packet forwarding decisions are based only on the destination (not source and destination) of the packet.

Our first result is that, for general preferences, computing an optimal set of routes is NP-hard; it is even NP-hard to compute a solution that approximates the optimum to within a factor of $O(n^{1/4-\epsilon})$, where n is the number of nodes in the network, and ϵ is an arbitrarily small positive constant. We prove this result by an approximability-preserving reduction from the Maximum Independent Set problem.

This leads us to consider a restricted class of utility functions that we call *next-hop preferences*. The restriction is that an AS's utility for a route can only depend on the first hop along that route. This class of utility functions captures preferences arising from customer/provider/peer relationships an AS might have with its neighbors. These commercial relationships are a major motivation for allowing flexible policy routing in BGP, and so this is an interesting class of preferences to study. We show that, for next-hop preferences, the welfare-maximization problem reduces to finding a maximum-weight directed spanning tree to each destination and is hence computable in polynomial time. We derive a strategyproof mechanism for this problem and show that it can also be computed in polynomial time.

We next ask whether it is possible to implement this mechanism with a distributed, BGP-based algorithm. Unfortunately, we find that this is not the case. In order to prove that a BGP-based implementation is impractical, we refine the model of BGP-based computation given in [77] and show that any implementation

of the welfare-maximizing policy-routing mechanism would be unacceptable, even on Internet-like graphs with small numeric valuations, for two reasons: (1) The selected routes may be long, and hence the routing algorithm may take a long time to converge; and (2) Any change in any AS's utilities may require communication to $\Omega(n)$ nodes, which defeats the rationale of using a path-vector protocol such as BGP. Thus, we conclude that, unlike the lowest-cost routing mechanism of [77], this mechanism is not easy to implement in the current Internet.

Mechanisms, and indeed Internet algorithms in general, need to be compatible with the existing protocols that they seek to extend or replace; this allows them to be adopted gradually. Positive results about protocol compatibility have been studied earlier, *e.g.*, in [77, 80]. However, proving negative results about protocol compatibility is more difficult; to our knowledge, we are the first to prove that a mechanism is *incompatible* with a given protocol. Thus, part of our contribution is refinement of the BGP-based computational model to allow negative results to be proven. Further, we believe that the "dynamic stability" criterion introduced here could potentially be used to prove hardness results for other Internet-algorithmic problems.

3.3 Autonomous nodes and distributed mechanisms

In this subsection we discuss the the work of Mitchell and Teague [138].

Standard distributed algorithmic mechanism design [17, 77, 78] uses a model that separates the computational entities implementing the algorithm from the strategic entities that provide its inputs. In the standard model, there may be several strategic agents who reside at each computational node in the network. The agents provide some input to the node, possibly lying, and the node then faithfully executes the algorithmic mechanism. This model reflects the most important characteristics of market situations, such as satellite television with tamper-resistant receivers in customer homes, in which the mechanism designer has complete control over the hardware and software used to implement the mechanism. However, the standard model omits an important aspect of user behavior in systems such as network routing in which autonomously administered routers may be configured in complex ways to serve the business objectives of their owners. In Internet routing, a system administrator may choose to modify router software if this improves local performance, regardless of whether the modification deviates from published Internet routing standards. Further, malicious or careless administrators may do so in ways that are not even beneficial to themselves (by standard measures). We consider the consequences of employing hardware or software that is controlled by strategic agents. We assume that strategic agents control the computation at each local node that implements part of a distributed algorithmic mechanism. For simplicity, and to aid comparison between our "SN" model and standard distributed algorithmic mechanism design, we investigate multicast cost sharing, a traditional problem with known "FPS" distributed solutions [78].

We use a network model that includes mostly selfish agents with some completely honest and a few malicious ones. We consider this a reasonable model for many Internet applications, possibly closer to reality than either the trusting distributed algorithmic mechanism design view or the security view that emphasizes worst-case scenarios. The vast majority of nodes on the Internet today are corporations that make rational decisions designed to maximize their profits. There are a few benevolent nodes (such as universities or government-subsidized sites) and a small number of actively malicious ones.

We focus on the example of the marginal cost mechanism for multicast cost sharing described by Feigenbaum *et al.* [78]. This mechanism shares the cost of a multicast transmission, such as a movie, among a tree of participating nodes. We may think of their distributed algorithm as being run by tamper-proof routers or set-top boxes. We will refer to this protocol as "the FPS protocol."

If the FPS protocol is implemented naively in the new model where the nodes can implement the algorithm of their choice, then selfish agents can benefit from cheats that are not possible in the model where

the nodes must execute the algorithm correctly. They can improve their welfare by telling different lies at different steps in the protocol, lying about what they have received from others or paying the incorrect amount at the end.

The multicast model includes a content provider that initiates the transmission and receives payments from each agent. This provides a convenient central point for providing further economic motivations to the agents. One method we use adds extra authentication to the protocol to allow each agent to "prove" that it behaved honestly. The content provider audits every agent with some probability and fines those who cannot produce a "proof". The content provider's computational burden and communication cost can be made constant by increasing the size of the fines with the size of the network. It must do a small number of local checks of one agent at a time. In other applications, where payments may not all reach a central point, we expect the same idea to apply in a more distributed fashion. We present two slightly different authenticated protocols, each with different messages and incentives. The first one is a lighter protocol whose main property is that honesty is an equilibrium—no agent is motivated to deviate from it if it believes that all the others will execute it correctly. Furthermore, as long as agents are selfish and keep to the protocol when there isn't a strictly better option, all agents execute the protocol correctly. The second protocol contains an additional signed message and is much stronger: we show that keeping to it is strictly more profitable than any alternative.

We examine security by introducing a malicious agent into the system and considering how much it can cause the mechanism to fall below the social optimum. We are also interested in how much such attacks cost, but do not expect the malicious agent to be rational. We show that the FPS scheme in its original model is quite secure against attack by a single agent, but that an unauthenticated implementation is not at all secure in the model where nodes can deviate from the protocol at will. We then show that our strongest authenticated scheme is almost as secure as forbidding the agents to deviate from the protocol, and that the presence of malicious nodes does not cause selfish ones to wish to deviate from the protocol. The only requirement is that all the malicious node's neighbors are honest.

4 Authorization management

A number of authorization management frameworks have been proposed in recent years, including PolicyMaker, its successor KeyNote [34], and the related infrastructures SDSI [166] and SPKI [73]. These systems go beyond simple certificate formats such as x.509 [108] by providing decentralized, policy-controlled authorization, with possibilities of permission updates, delegation and (in some systems) revocation. All of these frameworks are characterized by an algorithm. The algorithm does not lend itself to careful proofs of the properties of policies. For this it seems better to have a logic tailored to expressing properties of policies. Policies written in such a logic can then be proved correct, using techniques such as model checking.

It is clear that a rich logic will be required to deal with the notions like authentication and authorization. The first substantial, sustained effort to use formal logic as the foundation of a trust-management engine is Li's Delegation Logic (DL) [121]. DL is based on well-understood principles of logic programming and knowledge representation. Another effort [101] involves using pure first-order logic, augmented with a truth predicate. Yet other logics [1, 100] have been proposed for reasoning about one component of the puzzle—local and global names. More experience is required to ascertain to what extent any of these approaches can capture concerns of interest.

4.1 Using first-order logic to reason about policies

The results discussed in this subsection are due to Halpern and Weissman [102].

A policy describes the conditions under which an action, such as reading a file, is permitted or forbidden. Digital content providers have a rough idea of what their policies should be. Unfortunately, policies are typically described informally. As a result, their meaning and consequences are not always clear.

To better understand the problem, consider the statement "only librarians may edit the on-line catalog". We can view this statement as a policy, because it governs who may edit the catalog, based on whether or not the editor is a librarian. It is not clear if this policy permits librarians to make changes to the catalog or only forbids anyone who is not a librarian from doing so. The policy could be rewritten to remove this particular ambiguity, but others are likely to exist if policies are written in a natural language. Policy languages such as the Extensible rights Markup Language (XrML) [52] and Open Digital Rights Language (ODRL) [107] have the potential to be more formal (partly because their syntax is more restricted). Currently, however, the only semantics for these languages seems to be an English description of what the syntax means; thus, they also suffer from significant ambiguity. Our goal in this paper is to provide a logic with a clear syntax and semantics that can be used to represent and reason about policies. In addition, we want the logic to be well-suited to the needs of digital content providers. To achieve our objectives, we use a fragment of first-order logic. This automatically gives us a clear syntax and semantics; thus, it remains to argue that the logic is well-suited to the needs of digital content providers.

To be of practical use, a logic must satisfy (at least) the following three desiderata.

1. It must be expressive enough to capture in an easy and natural way the policies that people want to discuss.
2. It must be tractable enough to allow interesting queries about policies to be answered efficiently.
3. It must be usable by non-logicians, because we cannot expect policy makers and administrators to be well-versed in logic.

Of course, whether a logic is sufficiently expressive to meet our first objective depends very much on the application. To evaluate our approach, we gathered a large collection of policies from various libraries, including on-line collections, local and university libraries, the Library of Congress, and Cornell's Digital Library Research Group. We have written these policies in our language. In addition, we have begun to encode government policies in our language, including those that determine a person's eligibility for Social Security. Finally, we have created a translation from most of the XrML Core and all of the XrML Content Extension to our language. Details of the translation and a more complete discussion of the collected policies are given in a companion paper [102].

For the second desideratum, we focus on two key queries:

- Given a set of policies and an *environment* that provides all relevant facts (e.g., "Alice is a librarian", "Anyone who is a librarian for less than a year is a novice", etc.), does it follow that a particular action, such as Alice editing the on-line catalog, is permitted or forbidden?
- Is a set of policies consistent? In other words, are there no actions that are both permitted and forbidden by the policies in the set? This question is particularly interesting for collaboration. For example, suppose that Alice is writing the policies for her university's new outreach program. If the union of her policies and the university policies is consistent, then she knows that her policies do not contradict those of the university.

The answers to these questions could be used by enforcement mechanisms and individuals who want to do regulated activities. More importantly, we believe that the answers provide a reasonably good understanding of the policies, increasing our confidence that the formal statements capture the informal rules and the informal rules capture the policy creator's intent.

To address our third goal, the usability requirement, we developed, and are currently refining and extending, a prototype that allows users to enter policies, as well as facts about their environment, and to ask questions about them. This software will be tested by University of Virginia librarians as part of the Mellon-Fedora project [159] to verify that the language can be used by people who have not been trained in logic.

There have been a number of attempts to give formal semantics to policies, some of which involve first-order logic. Most of the first-order approaches are based on some variant of Datalog [86]. By beginning with Datalog, these solutions start with a language that is tractable, but not sufficiently expressive. They then extend the language to better meet the needs of applications. In particular, they find extensions that permit a limited use of negation and functions. The restrictions that we make are quite different from those made previously. We believe that the resulting language is especially well-suited for many applications, and has a number of advantages over variants of Datalog.

4.2 A formal foundation for XrML

Here we discuss further results of Halpern and Weissman [103].

The eXtensible rights Markup Language (XrML) is becoming an increasingly popular language in which to write software licenses. When first released in 2000, XrML received the support of many technology providers, content owners, distributors, and retailers, including Adobe Systems, Hewlett-Packard Laboratories, Microsoft, Xerox Corp., Barnesandnoble.com, and Time Warner Trade Publishing. Companies including Microsoft, OverDrive, and DMDsecure have publicly announced their agreement to build products and/or services that are XrML compliant. Currently, XrML is being used by international standard committees as the basis for application-specific languages that are designed for use across entire industries. For example, the Moving Picture Experts Group (MPEG) has selected XrML as the foundation for their MPEG-21 Rights Expression Language (MPEG-21 REL). (See http://www.xrml.org for more information.) It is clear that a number of industries are each moving towards a standard language for writing licenses, and that many of these standard languages are likely to be based on XrML. To understand the new standards, we need to understand XrML.

XrML does not have formal semantics. Instead, the XrML specification [146] presents the semantics in two ways. First is an English description of the language. Second is an English description of an algorithm that determines if a permission follows from a set of licenses. Unfortunately, the two versions of the semantics do not agree. To make matters worse, the algorithm has unintuitive consequences that do not seem to reflect the language writers' intent.

As a first step towards addressing these issues, we provide a formal semantics for XrML. To the best of our knowledge, we are the first to do this. We give the language formal semantics by providing a translation from XrML licenses to formulas in modal first-order logic. We verify the translation by proving that the algorithm included in the XrML document, slightly modified to correct the unintuitive behavior, matches our semantics. More precisely, the algorithm says that a permission follows from a set of licenses iff the translated permission is a logical consequence of the translated licenses. We then consider the complexity of determining if a permission is implied by a set of licenses. We show that the general problem is NP-hard, but, for an expressive fragment of the language, it is polynomial-time computable.

5 Protocols for diffuse infrastructure and secure communication

Diffuse computation requires new network protocols for locating and maintaining inventory of diffuse services and for secure and reliable communication among components of a diffuse system. Reliable, secure communication is a cornerstone of assured diffuse computation since distributed services must be able to communicate reliably in the face of network unreliability, network-based interference, or attack. Protocol design and analysis is a difficult problem. Some of the difficulties come from the uncertain nature of distributed computation, others from matters of scale and efficiency, and, for protocols with security properties, some difficulties arise from the subtleties of cryptographic primitives.

5.1 Robustness of class-based path-vector systems

The work described in this subsection is due to Jaggard and Ramachandran [111]. Their collaboration began while they were graduate students supported by this project, at the University of Pennsylvania and Yale, respectively.

The standard Internet inter-domain routing protocol, the Border Gateway Protocol (BGP), determines routes using independently configured policies in autonomously administered networks. Little global coordination of policies takes place between ASs, because (1) ASs are reluctant to reveal details about internal routing configuration, and (2) BGP contains no reliable mechanism to permanently attach information to a route as it is shared throughout the network. However, without global coordination, interaction of locally configured policies can lead to global routing anomalies [49, 96, 132, 183], *e.g.*, route oscillation and inconsistent recovery from link failures. Because the techniques used to configure policies and the protocol's specification have evolved separately, there is an inherent trade-off between, on one hand, maintaining rich semantic expressiveness available with current vendor-developed configuration languages and autonomy in policy configuration, and, on the other hand, guaranteeing that the protocol will converge robustly, *i.e.*, predictably, even in the presence of link and node failures. Griffin, Jaggard, and Ramachandran [95] showed that achieving all three of these design goals requires a non-trivial global constraint on the network, but they left open the question of how to identify and enforce the constraint.

We answer this question in the context of class-based path-vector systems. Path-vector systems, introduced in [95], provide a formal model for design and analysis of path-vector protocols and their policy-configuration languages. Class-based systems focus on a generalization of next-hop-preference routing, where routing policy for an AS can be defined by the relationships (commercial or otherwise) between it and its neighboring ASs. The canonical example of such a system is a simplified version of BGP that takes into account the economic realities of today's commercial Internet—that ASs are connected to their customers, providers, and peers and that there are preference guidelines used to decide between routes learned from neighbors of different classes. The scope of class-based systems, however, goes beyond this "Hierarchical BGP" system; the framework can also be used to build and analyze systems with complete autonomy and those that allow arbitrary next-hop preference routing. Furthermore, any protocol specification that can be described by a countable-weight, monotone path-vector algebra [177] can also be described by a class-based path-vector system [112].

We provide the best known robustness constraint for class-based systems. The constraint ensures the robustness of networks that satisfy it; it is in fact the best possible robustness guarantee because, in networks that do not satisfy it, some set of nodes may write policies that cause route oscillations. (Our proof of this constructs such policies.) We give an algorithm to generate the constraint given only the design specification of the system. We then provide centralized and distributed algorithms to check networks for violation of the constraint and discuss their applications, including how to use our results to check a network with arbitrary

next-hop preferences for potential bad interactions. The distributed algorithm reveals almost no private policy information, provides several options for correcting a constraint violation, and has constant message complexity per link and limited storage at each node. We compare and contrast our algorithms with those in previous work.

Although it may be sufficient to provide a supplementary protocol enforcing some global conditions (and, indeed, our distributed algorithm, modified for BGP, can be run alongside BGP to detect potentially bad policy interactions), there are several benefits to this approach of analyzing robust protocol convergence from a design-framework perspective. First, the our algorithms preclude all policy-based oscillations in advance; as long as the constraint is enforced, the protocol can safely run on any network. Second, the approach is an integral part of designing policy-configuration languages. The design framework identifies provably sufficient local and global conditions needed for a protocol to achieve its design goals. We give the precise trade-off between the strength of local policy guidelines built into the policy-configuration language and the strength of the global assumption needed in the broad class-based context. The designer can use these results to consider what balance between local and global enforcement is desired and can incorporate the guidelines generated by our results into the design of multiple high-level policy languages—all before running the protocol on an actual network.

5.2 A formal analysis of some properties of Kerberos 5

The work discussed in this subsection is due to

Butler, Cervesato, Jaggard, and Scedrov

[38, 39]

Kerberos [117, 149, 148, 150] is a widely deployed protocol, designed to repeatedly authenticate a client to multiple application servers based on a single login. The protocol uses various credentials (tickets), encrypted under a server's key and thus opaque to the client, to authenticate the client to the server; this allows the client to obtain additional credentials or to request service from an application server. A formalization of Kerberos 4, the first publicly released version of this protocol, was given in [31] and has since been extended and thoroughly analyzed using an inductive approach [27, 28, 29, 30]. This analysis, through heavy reliance on the Isabelle theorem prover, yielded formal correctness proofs for a fairly detailed specification, and also highlighted a few minor problems. A simple fragment of the latest version, Kerberos 5, has been investigated using the state exploration tool Murφ [139]. This approach proved effective for finding an attack, which the authors of [139] note is unrealizable in a full implementation of Kerberos 5, but came short of proving positive correctness results.

Here we report on a project whose goal is to use the Multi-Set Rewriting (MSR) framework [70] to give a precise specification of Kerberos 5 at various levels of detail, ranging from a minimal account, similar to that used in [139], to a detailed formalization of every behavior encompassed by this complex suite [117, 148]. Our particular objectives include giving a precise and unambiguous description of this protocol, making its operational assumptions explicit, stating the properties it is supposed to satisfy, and proving that it satisfies these properties. This will complement the currently spotty and often vague information in the literature. This project is also intended as a test-bed for the MSR formalism [70] on a real-world protocol: we are interested in how easy it is to write large specifications in MSR, in what ways this language can be improved, and whether the insight gained with toy protocols scales up. In this work we have also started exploring forms of reasoning that best take advantage of the linguistic features of MSR.

We provide three formalizations of Kerberos 5, which we call our A, B, and C level formalizations. The B and C level formalizations add detail to the A level formalization but are not otherwise related. The

A level formalization omits most timestamps and all optional features, including only what we believe is needed to provide authentication. It is similar to the formalization of Kerberos 4 in [27, 28, 29], but without timestamps. This level of abstraction is a good starting point to utilize the proof techniques demonstrated, providing a formalization which is not overly complicated (making proofs feasible), but which retains many properties of the full Kerberos 5 protocol. Our B level formalization adds some timestamps and temporal checks to our A level formalization, thus closely paralleling the formalization of Kerberos 4 in [27, 28, 29]. We have not found any new and interesting properties or anomalies related to the timestamps here; the two features of the B level which are not found in [27, 28, 29]—the single option of mutual authentication and error messages—seemed like the most promising area to focus our efforts. This leads to our C level formalization, which does not include temporal checks or most timestamps. It extends the A level formalization by making mutual authentication optional and adding error messages, along with several low-level aspects of the protocol, namely options, flags, and checksums, none of which has appeared in any previous study of Kerberos. We have focused our investigations on the A and C level formalizations, with the abstraction of the former facilitating reasoning about the protocol and the detail of the latter providing an interesting step on the way to formalizing the protocol in full detail.

We have proved confidentiality and authentication properties [93] for our A level formalization, and have extended some of these proofs to our C level formalization; in each case, we use the notion of rank and corank functions, inspired by [171]. While Kerberos specifically disclaims responsibility for preventing denial of service attacks, we have noticed instances of other potentially curious protocol behavior. The first, which arises in both the A level and C level formalizations, violates properties that were proved to hold for Kerberos 4 [27] and highlights the structural differences between the messages in versions 4 and 5 of the protocol. The other three instances of curious behavior, seen only in our C level formalization, take advantage of protocol options available at this level; the first and third of these are related to the behavior also seen at the A level, while the second is completely unrelated. Our informal analysis of the B level formalization did not reveal any new anomalies.

5.3 Abstraction and refinement in protocol derivation

The results reported in this subsection are due to Datta, Derek, Mitchell, and Pavlovic [59].

Many network protocols with security objectives are designed using a smaller set of common protocol concepts, such as challenge-response, Diffie-Hellman-like key agreement, and “cookies” to reduce potential denial of service. In previous work [56, 57, 58], we proposed a protocol derivation framework, based on the use of composition, refinement, and transformation, and a formal logic for stating and proving properties of protocols. In this framework, a protocol designer may choose two initial protocol components, refine each of them, compose the results to get a candidate protocol, then apply one or more transformations to improve efficiency or resist particular forms of attack. While properties of the resulting protocol may be proved formally in our logic, the structure of protocol proofs we have previously devised have not always followed the structure of the protocol derivation. We extend the previous protocol logic with higher-order features, making it possible to define protocol templates and reason about their instances. Using protocol templates, we are able to characterize the correctness properties of a class of protocol refinements, furthering our long-term effort toward a framework for systematically deriving verified security protocols.

Composition combines separate protocols, refinements change the content or structure of individual messages, and transformations alter the structure of a protocol. In our formal logic, we may prove properties about a composed protocol from its parts, using a set of composition inference rules [57, 58]. The composition rules involve local reasoning about steps in each role and global reasoning about invariants in the protocol or set of protocols in use. In a protocol refinement, a message or portion of a message

is systematically refined by, for example, adding additional data or otherwise changing the data contained in one or more messages. For example, replacing a plaintext nonce by an encrypted nonce, in both the sending and receiving protocol roles, is a protocol refinement. While refinements seem to arise naturally in contemporary practical protocols [6, 118], they provide more of a challenge for formal reasoning. One reason is that refinements may involve replacement, and replacement of one expression by another does not have a clean formulation in standard mathematical logic. This immediate problem is solved by introducing protocol templates and decomposing term replacement into an abstraction step of selecting an appropriate template and an instantiation step that replaces template variables with protocol expressions. Another issue, addressed by associating hypotheses with a proof about a template, is that a protocol refinement may not be useful for all protocols, but only for protocols that satisfy certain hypotheses.

To give a simple example, suppose we have a protocol containing messages that use symmetric encryption, and suppose that some useful property of this protocol is preserved if we replace symmetric encryption by use of a keyed hash. We can capture the relationship between these two protocols by writing an "abstract" protocol template with function variables in the positions occupied by either encryption or keyed hash. Then the two protocols of interest become instances of the template. In addition, a similar relationship often works out for protocol proofs. If we start with a proof of some property of the protocol that contains symmetric encryption, some branches of the proof tree will establish properties of symmetric encryption that are used in the proof. If we replace symmetric encryption by a function variable, then the protocol proof can be used to produce a proof about the protocol template containing function variables. This is accomplished by replacing each branch that proves a property of symmetric encryption by a corresponding hypothesis about the function variable. Once we have a proof for the protocol template obtained by abstracting away the specific uses of symmetric encryption, we can consider replacing the function variable with keyed hash. If keyed hash has the properties of symmetric encryption that were used in the initial proof, we can use proofs of these properties of keyed hash in place of the assumptions about the function variable. Thus an abstraction step and an instantiation step bring us both from a protocol with symmetric encryption to a protocol with keyed hash, and from a proof of the initial protocol to a proof of the final one. The role of the protocol template in this process is to provide a unified proof that leads from shared properties of two primitives (symmetric encryption or keyed hash) to a protocol property that holds with either primitive.

After describing the formal framework, we illustrate the use of protocol templates with several examples. As an example of multiple instantiations of a single template, we prove an authentication property of a generic challenge-response protocol, and then show how to instantiate the template to ISO-9798-2, ISO-9798-3, or SKID3. As an example of one protocol that is an instance of two templates, we show how to reason about an identity-protection refinement using an authentication template and an encryption template. The third example compares two authentication protocol templates, one that can be instantiated to the ISO-9798 family of protocols, and one that can be instantiated to STS and SIGMA. The first reflects the authentication mechanism used in JFKi, while the second corresponds to IKE, JFKr, and IKEv2 authentication. While there has been considerable debate and discussion in the IETF community about the tradeoffs offered by these two protocols, previous analyses are relatively low-level and do not illustrate the design principles involved. However, it is possible to compare the authentication properties of the two approaches by comparing the templates.

While our past work on protocol derivation has given rational reconstructions of known protocols, we can also use protocol derivation to combine known protocols in new ways. We begin with two separate derivations. The first, within the JFK family, starts with Diffie-Hellman key exchange protocol and a basic three-step challenge-response protocol. These two are combined to form Station-to-Station (STS), whose key secrecy is based on Diffie-Hellman and authentication property is based on challenge-response. A few steps from STS bring us to a form of JFK. An orthogonal derivation begins with Diffie-Hellman and modifies the functions used, through MTI/A [131] and UM [15] to reach MQV [119]. These two protocol derivations

provide a two-dimensional matrix of protocols that have not been explored, to our knowledge. The most sophisticated is a form of JFK using MQV in place of Diffie-Hellman as its key-exchange component. This protocol provides forms of key secrecy, mutual authentication, forward secrecy, known-key security, computational efficiency, identity protection, and denial-of-service protection, inheriting these qualities from protocol design patterns used to produce the protocol.

There are several differences between the work described in this subsection and some other protocol analysis efforts. To begin with, our basic model of protocol execution and possible attacker actions is the traditional "Dolev-Yao model" [65, 147] that has been used in many other efforts [114, 139, 135, 158, 169]. In particular, the protocol refinements we consider all replace symbolic operations of one form with symbolic operations of another; we do not consider "refining" a symbolic operation on message strings to a computable operation on bit sequences. While it is an important research direction to relate our model to computational models such as [42, 21, 163], we currently believe that "computational soundness" of symbolic methods is a separable goal that will lead to greater use of the kind of logical methods considered in this subsection. At a more detailed level, there are some important differences between the way that we reason about incremental protocol construction and alternative approaches such as "universal composability" [42]. In universal composability, properties of a protocol are stated in a strong form so that the property will be preserved under a wide class of composition operations. In contrast, our protocol proofs proceed from various assumptions, including invariants that are assumed to hold about any environment in which the protocol operates. The ability to reason about protocol parts under assumptions about the way they will be used offers greater flexibility and appears essential for developing modular proofs about certain classes of protocols.

5.4 A probabilistic polynomial-time calculus for the analysis of cryptographic protocols

We begin this subsection with a discussion of the papers [140] by Mitchell, Ramanathan, Scedrov, and Teague and [129] by Mateus, Mitchell, and Scedrov.

A variety of methods are used for analyzing and reasoning about security protocols. The main systematic or formal approaches include specialized logics such as BAN logic, [37], special-purpose tools designed for security protocol analysis, [114], and theorem proving [158] and model-checking methods using several general purpose tools. described in [169, 125, 134, 139, 167, 170].

Although such methods differ in significant ways, many of them reflect the same basic assumptions about the way an adversary may interact with the protocol or attempt to decrypt encrypted messages . In the common model, largely derived from [66] and suggestions found in [147], a protocol adversary is allowed to choose among possible actions nondeterministically. This is a convenient idealization, intended to give the adversary a chance to find an attack if there is one. In the presence of nondeterminism, however, the set of messages an adversary may use to interfere with a protocol must be restricted severely. Although the idealized assumptions make protocol analysis tractable, they also make it possible to "verify" protocols that are in fact susceptible to simple attacks that lie outside the adversary model. Another limitation is that a deterministic or nondeterministic setting does not allow us to analyze probabilistic protocols. In other words, actual protocols use actual cryptosystems that may have their own weaknesses, or might employ probabilistic techniques not expressed in the idealized model.

Recently there have been several research efforts to relate the idealized model to cryptographic techniques and the computational model based on probabilistic polynomial-time computation, including [41, 123, 140, 161, 3, 42, 21, 22]. While these efforts develop rigorous mathematical settings carried out so far only "by hand", it is hoped that they will eventually lead to a new generation of "high fidelity" automated tools for security analysis that will be able to express the methods and concepts of modern cryptography.

Our initial contribution to this line of research was a formulation of a process calculus proposed in [123, 140] as the basis for a form of protocol analysis that is formal, yet closer in foundations to the mathematical setting of modern cryptography. The framework relies on a language for defining communicating polynomial-time processes. The reason we restrict processes to probabilistic polynomial time is so that we can reason about the security of protocols by quantifying over all "adversarial" processes definable in the language. In effect, establishing a bound on the running time of an adversary allows us to relax the simplifying assumptions. Specifically, it is possible to consider adversaries that might send randomly chosen messages, or perform sophisticated (yet probabilistic polynomial-time) computation to derive an attack from messages it overhears on the network. An important aspect of our framework is that we can analyze probabilistic as well as deterministic encryption functions and protocols. Without a probabilistic framework, it would not be possible to analyze an encryption function such as [71], for which a single plaintext may have more than one ciphertext.

Some of the basic ideas of our prior work are presented in [123, 140]. Further example protocols are considered in [124]. The closest technical precursor is [2], which uses observational equivalence and channel abstraction but does not involve probability or computational complexity bounds. Prior work on CSP and security protocols, *e.g.*, [167, 170], also uses process calculus and security specifications in the form of equivalence or related approximation orderings on processes.

This approach is based on the intuition that security properties of a protocol P may be expressed by means of existence of an idealized protocol Q such that for any adversary M, the interactions between M and P have the same observable behavior as the interactions between M and Q. The idea of expressing security properties in terms of some comparison to an ideal protocol goes back at least to [91, 26, 25, 136]. Here we emphasize a formalization of this idea by using observational equivalence, a standard notion from programming language theory. That is, two protocols P and Q are observationally equivalent if any program $C[P]$ has the same observable behavior as the program $C[Q]$, with Q instead of P. The reason observational equivalence is applicable to security analysis is that it involves quantifying over all possible additional processes represented by the contexts $C[\ ]$ that might interact with P and Q, in precisely the same way that security properties involve quantifying over all possible adversaries. Our framework is a refinement of this approach. In our asymptotic formulation [123, 140], observational equivalence between probabilistic polynomial-time processes coincides with the traditional notion of indistinguishability by polynomial-time statistical tests [89, 188], a standard way of characterizing cryptographic primitives.

In this paper we derive a compositionality property from inherent structural properties of our process calculus. Basically, compositionality states that composing secure protocols remains secure. We obtain a general result of this kind in two steps. We consider a notion of a secure realization, or, *emulation* of an ideal protocol, motivated by [41] but here expressed by means of asymptotic observational equivalence. We show that the notion of emulation is congruent with the primitives of the calculus. Compositionality follows because the security requirements are expressed in the form that a real protocol securely realizes an ideal protocol.

We also illustrate some of these concepts on a traditional cryptographic example of oblivious transfer [162, 74, 90, 43]. We show how the natural security requirements may be expressed in our calculus in the form that a real protocol emulates an ideal protocol. Finally, we establish an important relationship between the process calculus framework and the interactive Turing machine framework discussed in [42, 43, 161, 21, 22]. Indeed, the work based on [42, 21] provides an encyclopedic treatment of a number of security requirements in a compositional setting. However, the framework of interactive Turing machines, even if optimal to deal with complexity results, is rather low-level and does not seem naturally suited for specification of and reasoning about cryptographic protocols. Moreover, the framework of interactive Turing machines comes about from the connections between cryptography and complexity, and therefore, some

effort must be spent to obtain structural results, such as the composition theorem.

Subsequent work on unifying equivalence-based definitions of protocol security by Datta, Küsters, Mitchell, Ramanathan, Shmatikov [60] relates several research efforts that have led to three different ways of specifying protocol security properties by simulation or equivalence. Abstracting the specification conditions away from the computational frameworks in which they have been previously applied, it is shown that when asynchronous communication is used, universal composability [42], black-box simulatability [21], and process equivalence [140, 129] express the same properties of a protocol. Further, the equivalence between these conditions holds for any computational framework, such as process calculus, that satisfies certain structural properties. Similar but slightly weaker results are achieved for synchronous communication.

5.5 Contract-signing protocols

This section begins with a discussion of results due to Chadha, Mitchell, Scedrov, and Shmatikov [47, 45].

A variety of contract-signing protocols have been proposed in the literature, including gradual-release two-party protocols [32, 35, 164] and fixed-round protocols that rely on an adjudicating "trusted third party" [19, 20, 84, 127, 189]. In this subsection, we focus on fixed-round protocols that use a trusted third party optimistically, meaning that when all goes well, the third party is not needed. The reason for designing optimistic protocols is that if a protocol is widely or frequently used by many pairs of signers, the third party may become a performance bottleneck. Depending on the context, seeking resolution through the third party may delay termination, incur financial costs, or raise privacy concerns. Obviously, the value of an optimistic protocol, as opposed to one that requires a third party signature on every transaction, lies in the frequency with which "optimistic" signers can complete the protocol without using the third party.

Some useful properties of contract-signing protocols are *fairness*, which means that either both parties get a signed contract, or neither does, and *timeliness*, which generally means that each party has some recourse to avoid unbounded waiting. The reason for using a trusted third party in fixed-round protocols is a basic limitation [75, 153] related to the well-known impossibility of distributed consensus in the presence of faults [81]: no fixed-length two-party protocol can be fair. Although there is a trivial protocol with a trusted third party, in which both signers always send their signatures directly to it, protocols that are fair, timely, and usefully minimize demands on the third party have proven subtle to design and verify.

We refine previous models, formalizes properties from the literature on fixed-round two-party contract-signing protocols, and establishes relationships between them. We use the set-of-traces semantics for protocols, defining each instance of the protocol as the set of all possible execution traces, arranged in a tree. The set of traces of a protocol is derived from a multiset rewriting [70] presentation of the protocol, for concreteness, although other formalisms for characterizing protocols and their sets of traces would give similar results.

Model for optimism. One modeling innovation is an *untimed* nondeterministic setting that provides a set-of-traces semantics for optimism. Intuitively, optimistic behavior in contract signing is easily described as a temporal concept: an optimistic signer is one who waits for some period of time before contacting the trusted third party. If Alice is optimistic, and Bob chooses to continue the protocol by responding to Alice, then Alice will deliberately wait for Bob's message rather than contact the third party. Since the value of an optimistic protocol lies in what it offers to an optimistic player, we evaluate protocols subject to the assumption that one of the players follows an optimistic strategy. As a direct way of mathematically characterizing the sequence of actions that occur in optimistic play, we allow an optimistic player to deliberately give his opponent control over whether the optimist waits for a message. In other words, an optimistic player wishes to wait for a message. We allow an optimistic player to act on this wish by entering a waiting state until the opponent's move places the optimistic player in a non-waiting state. This gives us a direct way of defining

the set of traces associated with an optimistic signer, while staying within the traditional nondeterministic, untimed setting.

Impossibility result. In evaluating protocol performance for optimistic players, we prove that in every fair, timely protocol, an optimistic player suffers a disadvantage. The importance of this result is that optimistic protocols are only useful to the extent that signers may complete the protocol optimistically without contacting the third party. In basic terms, our theorem shows that whenever a protocol allows signers to avoid the third party, an optimistic signer gives the other signer unilateral control over the outcome at some point in the execution of the protocol.

To illustrate by example, consider an online stock-trading protocol with signed contracts for each trade. Suppose the broker starts the protocol, sending her commitment to sell stock to the buyer at a specific price, and the buyer responds with his commitment. To ensure timely termination, the broker also enjoys the ability to abort the exchange by contacting the trusted third party (TTP) if the buyer has not responded. Once the buyer commits to the purchase, he cannot use the committed funds for other purposes. Even if he has the option to contact the TTP immediately, an optimistic buyer will wait for some period of time for the broker to respond, hoping to resolve the transaction amicably and avoid the extra cost or potential delay associated with contacting the TTP. This waiting period may give the broker a useful window of opportunity. Once she has the buyer's commitment, the broker can wait to see if shares are available from a selling customer at a matching or lower price. The longer the buyer is inclined to wait, the greater chance the broker has to pair trades at a profit. If the broker finds the contract unprofitable, she can abort the transaction by falsely claiming to the TTP that the buyer has not responded. This broker strategy succeeds in proportion to the time that the buyer optimistically waits for the broker to continue the protocol; this time interval, if known exactly or approximately, gives the broker a period where she can decide *unilaterally* whether to abort or complete the exchange.

Since our main result only involves one run of an arbitrary contract-signing protocol, we do not need to consider sequences of protocol runs, or interleaving of concurrent runs.

We conclude this subsection with a brief discussion of some recent results of Chadha, Kremer, and Scedrov [46].

A finite-sate tool, MOCHA, is used to analyze the multi-party contract-signing protocols of Garay and MacKenzie (GM) [85] and of Baum-Waidner and Waidner (BW) [24]. MOCHA which allows specification of protocol properties in a branching-time temporal logic with game semantics. While the analysis does not reveal any errors in the BW protocol, in the GM protocol, serious problems with fairness for four signers and an oversight regarding abuse-freeness for three signers were discovered. A complete revision of the GM subprotocols is proposed in order to restore fairness.

6 Privacy and anonymity

6.1 Anonymity and information hiding in multiagent systems

The work discussed in this subsection is due to Halpern and O'Neill [98, 104]

Our primary goal is to provide a formal framework for reasoning about anonymity in multiagent systems. The importance of anonymity has increased over the past few years as more communication passes over the Internet. Web-browsing, message-sending, and file-sharing are all important examples of activities that computer users would like to engage in, but may be reluctant to do unless they can receive guarantees that their anonymity will be protected to some reasonable degree. Systems are being built that attempt to implement anonymity for various kinds of network communication (see, for example, [184, 88, 120, 165,

175, 179]). It would be helpful to have a formal framework in which to reason about the level of anonymity that such systems provide.

We view anonymity as an instance of a more general problem: information hiding. In the theory of computer security, many of the fundamental problems and much of the research has been concerned with the hiding of information. Cryptography, for instance, is used to hide the contents of a message from untrusted observers as it passes from one party to another. Anonymity requirements are intended to ensure that the identity of the agent who performs some action remains hidden from other observers. Noninterference requirements essentially say that *everything* about classified or high-level users of a system should be hidden from low-level users. Privacy is a catch-all term that means different things to different people, but it typically involves hiding personal or private information from others.

Information-hiding properties such as these can be thought of as providing answers to the following set of questions:

- What information needs to be hidden?
- Who does it need to be hidden from?
- How well does it need to be hidden?

By analyzing security properties with these questions in mind, it often becomes clear how different properties relate to each other. These questions can also serve as a test of a definition's usefulness: an information-hiding property should be able to provide clear answers to these three questions.

In earlier work [97], we formalized secrecy in terms of knowledge. Our focus was on capturing what it means for one agent to have *total* secrecy with respect to another, in the sense that no information flows from the first agent to the second. Roughly speaking, a high-level user has total secrecy if the low-level user never knows anything about the high-level user that he didn't initially know. Knowledge provides a natural way to express information-hiding properties—information is hidden from a if a does not know about it. Not surprisingly, our formalization of anonymity is similar in spirit to our formalization of secrecy. Our definition of secrecy says that a classified agent maintains secrecy with respect to an unclassified agent if the unclassified agent doesn't learn any new fact that depends only on the state of the classified agent. That is, if the agent didn't know a classified fact ϕ to start with, then the agent doesn't know it at any point in the system. Our definitions of anonymity say that an agent performing an action maintains anonymity with respect to an observer if the observer never learns certain facts having to do with whether or not the agent performed the action.

Obviously, total secrecy and anonymity are different. It is possible for i to have complete secrecy while still not having very strong guarantees of anonymity, for example, and it is possible to have anonymity without preserving secrecy. However, thinking carefully about the relationship between secrecy and anonymity suggests new and interesting ways of thinking about anonymity. More generally, formalizing anonymity and information hiding in terms of knowledge is useful for capturing the intuitions that practitioners have.

We are not the first to use knowledge and belief to formalize notions of information hiding. Glasgow, MacEwen, and Panangaden [87] describe a logic for reasoning about security that includes both *epistemic* operators (for reasoning about knowledge) and *deontic* operators (for reasoning about permission and obligation). They characterize some security policies in terms of the facts that an agent is permitted to know. Intuitively, everything that an agent is not permitted to know must remain hidden. Our approach is similar, except that we specify the formulas that an agent is *not* allowed to know, rather than the formulas she is permitted to know. One advantage of accentuating the negative is that we do not need to use deontic operators in our logic.

Epistemic logics have also been used to define information-hiding properties, including noninterference and anonymity. Gray and Syverson [94] use an epistemic logic to define probabilistic noninterference, and Syverson and Stubblebine [180] use one to formalize definitions of anonymity. The thrust of our work is quite different from these. Gray and Syverson focus on one particular definition of information hiding in a probabilistic setting, while Syverson and Stubblebine focus on describing an axiom system that is useful for reasoning about real-world systems, and on how to reason about and compose parts of the system into adversaries and honest agents. Our focus, on the other hand, is on giving a semantic characterization of anonymity in a framework that lends itself well to modeling systems.

Shmatikov and Hughes [106] position their approach to anonymity (which is discussed in more detail in Section 5.3 as an attempt to provide an interface between logic-based approaches, which they claim are good for specifying the desired properties (like anonymity), and formalisms like CSP, which they claim are good for specifying systems. We agree with their claim that logic-based approaches are good for specifying properties of systems, but also claim that, with an appropriate semantics for the logic, there is no need to provide such an interface. While there are many ways of specifying systems, many end up identifying a system with a set of runs or traces, and can thus be embedded in the runs and systems framework that we use.

Definitions of anonymity using epistemic logic are *possibilistic*. Certainly, if j believes that any of 1000 users (including i) could have performed the action that i in fact performed, then i has some degree of anonymity with respect to j. However, if j believes that the probability that i performed the action is .99, the possibilistic assurance of anonymity provides little comfort. Most previous formalizations of anonymity have not dealt with probability; they typically conclude with an acknowledgment that it is important to do so, and suggest that their formalism can indeed handle probability. One significant advantage of our formalism is that it is completely straightforward to add probability in a natural way, using known techniques [99]. This lets us formalize the (somewhat less formal) definitions of probabilistic anonymity given by Reiter and Rubin [165].

We are more concerned with defining and specifying anonymity properties than with describing systems for achieving anonymity or with verifying anonymity properties. We want to define what anonymity means by using syntactic statements that have a well-defined semantics. Our work is similar in spirit to previous papers that have given definitions of anonymity and other similar properties, such as the proposal for terminology given by Pfitzmann and Köhntopp [160] and the information-theoretic definitions of anonymity given by Diaz, Seys, Claessens, and Preneel [61] and Serjantov and Danezis [173].

6.2 Economics of anonymity

In this subsection we discuss the work due to Acquisti, Dingledine, and Syverson [4]. Dr. Syverson of the Naval Research Laboratory is an external collaborator of the project.

Individuals and organizations need anonymity on the Internet. People want to surf the Web, purchase online, and send email without exposing to others their identities, interests, and activities. Corporate and military organizations must communicate with other organizations without revealing the existence of such communications to competitors and enemies. Firewalls, VPNs, and encryption cannot provide this protection; indeed, Diffie and Landau have noted that traffic analysis is the backbone of communications intelligence, not cryptanalysis [62].

With so many potential users, it might seem that there is a ready market for anonymity services—that is, it should be possible to offer such services and develop a paying customer base. However, with one notable exception (the Anonymizer [16]) commercial offerings in this area have not met with sustained success. We could attribute these failures to market immaturity, and to the current economic climate in general. However,

this is not the whole story.

We explore the incentives of participants to offer and use anonymity services. We set a foundation for understanding and clarifying our speculations about the influences and interactions of these incentives. Ultimately we aim to learn how to align incentives to create an economically workable system for users and infrastructure operators.

We show that, contrary to usual assumptions, infrastructures shared between participants with diverse incentives can have equilibria at an optimal level of free-riding. Participants with high sensitivity to traffic analysis have an incentive to provide anonymity service to those with low sensitivity. We also describe the exit node liability problem in which much of the liability cost of traffic analysis resistant communication may be born by system exit nodes. This has already caused a change in the default exit policy for nodes in the Tor network [63], currently the most widely deployed anonymous infrastructure for general Internet communication. Our analysis has also resulted in a change to the recruitment strategy for nodes in the Tor network and the Mixminion network.

6.3 Universal re-encryption for mixnets

In this subsection we discuss work due to Golle, Jakobsson, Juels, and Syverson [92].

A *mix network* or *mixnet* is a cryptographic construction that invokes a set of servers to establish private communication channels [48]. One type of mix network accepts as input a collection of ciphertexts, and outputs the corresponding plaintexts in a randomly permuted order. The main privacy property desired of such a mixnet is that the permutation matching inputs to outputs should be known only to the mixnet, and no one else. In particular, an adversary should be unable to guess which input ciphertext corresponds to an output plaintext any more effectively than by guessing at random.

One common variety of mixnet known as a *re-encryption mixnet* relies on a public-key encryption scheme, such as that of ElGamal [83], which allows for re-encryption of ciphertexts. For a given public key, a ciphertext C' is said to represent a re-encryption of C if both ciphertexts decrypt to the same plaintext. In a re-encryption mixnet, the inputs are submitted encrypted under the public-key of the mixnet. (The corresponding private key is held in distributed form among the servers.) The batch of input ciphertexts is processed sequentially by each mix server. The first server takes the set of input ciphertexts, re-encrypts them, and outputs the re-encrypted ciphertexts in a random order. Each server in turn takes the set of ciphertexts output by the previous server, and re-encrypts and mixes them. The set of ciphertexts produced by the last server may be decrypted by a quorum of mix servers to yield plaintext outputs. Privacy in this mixnet construction der! ives from the fact that the ciphertext pair (C, C') is indistinguishable from a pair (C, R) for a random ciphertext R to any adversary without knowledge of the private key.

In this paper, we propose a new type of public-key cryptosystem that permits *universal re-encryption* of ciphertexts. We introduce the term universal encryption to mean re-encryption without knowledge of the public key under which a ciphertext was computed. Like standard re-encryption, universal re-encryption transforms a ciphertext C into a new ciphertext C' with same corresponding plaintext. The novelty in our proposal is that re-encryption neither *requires* nor *yields* knowledge of the public key under which a ciphertext was computed.

When applied to mix networks, our universal re-encryption technique offers new and interesting functionality. Most importantly, mix networks based on universal re-encryption dispense with the cumbersome protocols that traditional mixnets require in order to establish and maintain a shared private key. We discuss more benefits and applications of universal mixnets in the next section. It is possible to construct a *universal mixnet* based on universal re-encryption roughly as follows. Every input to the mixnet is encrypted under the

public key of the recipient for whom it is intended. Thus, unlike standard re-encryption mixnets, universal mixnets accept ciphertexts encrypted under the individual public keys of receivers, rather than encrypted under the unique public key of the mix network. These ciphertexts are universally re-encrypted and mixed by each server. The output of a universal mixnet is a set of ciphertexts. Recipients can retrieve from the set of output ciphert! exts those addressed to them, and decrypt them.

6.4 Synchronous batching: From cascades to free routes

In this subsection we discuss work due to Dingledine, Shmatikov, and Syverson [64].

Modern deployed mix networks, including Mixmaster [143] and its successor Mixminion [55], are subject to partitioning attacks: a passive adversary can observe the network until a target message happens to stand out from the others [33], and an active adversary can manipulate the network to separate one message from the others via blending attacks [174]. Berthold et al. argue [33] that partitioning opportunities arise because the networks use a *free-route* topology—one where the sender can choose the mixes that make up her message's path. They suggest instead a *cascade network* topology, where all senders choose from a set of fixed paths through the mix network.

We argue that the cascade design resolves these attacks because it uses a *synchronous batching* strategy, not because it uses a particular network topology. We show that synchronous batching prevents these attacks even when free routes are used. Further, we explore three topologies with synchronous batching—cascades, stratified (a restricted-route hybrid topology), and free-route—and find that the free-route network provides the highest expected anonymity as well as the best robustness to node failure.

6.5 Negotiated Privacy

The work discussed in this subsection is due to Jarecki, Lincoln, and Shmatikov [113].

Exponential growth in digital information gathering, storage, and processing capabilities inexorably leads to conflict between well-intentioned government or commercial datamining, and fundamental privacy interests of individuals and organizations. We propose a mechanism that provides cryptographic fetters on the mining of personal data, enabling efficient mining of previously-negotiated properties, but preventing any other uses of the protected personal data. Our approach does not rely on complete trust in the analysts to use the data appropriately, nor does it rely on incorruptible escrow agents. Instead, we propose conditional data escrow where the data generators, not the analysts, hold the keys to the data, but analysts can verify that the pre-negotiated queries are enabled. Our solution relies on verifiable, anonymous, and deterministic commitments which play the role of tags that mark encrypted entries in the analyst's database. The database owner cannot learn anything from the encrypted entries, or even verify his guess of the plaintext on which these entries are based. On the other hand, the verifiable and deterministic property ensures that the entries are marked with consistent tags, so that the database manager learns when the number of entries required to enable some query reaches the pre-negotiated threshold.

7 Networking

The SwitchWare active-network architecture (see `www.cis.upenn.edu/~switchware/`) is ideally suited to the study of market-based diffuse computation. In essence, the active networking proposal is that user-defined programs, perhaps limited by security considerations and pre-arranged trust relationships, may

be loaded into network elements such as routers. The loaded code can provide new services, in particular those that require algorithmic control rather than simple parametric control. The SwitchWare active-networking platform has been implemented using the Caml typesafe programming language (a functional programming language in the style of ML) and provides extensive support for developing networked applications. Among the applications developed for SwitchWare are an active firewall, an active internetwork, and an active bridge. The programming environment is most mature on the Linux operating system, but a version has been created on the Nemesis operating system (RCANE), and we believe that ports to BSD-based Unixes such as OpenBSD are straightforward. A variant of the system, called the Secure Active Networking Environment (SANE), provides extensive support for cryptographic protocol development, supporting the investigation of security and privacy issues in the diffuse computing.

A specific example of the use of the active networking technology is the use of the programmable nodes to support agents with their individual valuation functions and market strategies. Agents can be implemented in the nodes, as can network algorithms for interconnecting market participants (*e.g.*, auction-support infrastructure deletes irrelevant bids on their way to a seller). Agents can use active packets containing code to implement policies that may require information that is local and dynamic (such as resource prices on a node). For example, an active extension implementing an agent might highly value responsiveness in a distributed execution environment. Thus, for this agent, an active-packet program could examine measures of load, such as packet or process queues, to determine whether it should wait for service or migrate to another server. In this way, robustness and agent goals can be achieved with local autonomy while global mechanisms are applied.

Market-based control paradigms have been used extensively in computer systems design to deal with unpredictable resource demands, large system dynamics, and evolving behavior and priorities in control. An early example is the futures market in computer time designed by Sutherland [178]. A set of articles edited by Clearwater [50] covers the use of market-based control in various tasks such as memory allocation and network bandwidth management. Clearwater's collection is dominated by distributed systems and systems involving networks, because they exhibit dynamics, evolving behavior, and decentralized control (decentralized control and local autonomy are important factors in achieving scalability). Waldspurger *et al.* [185] used these techniques to create a computational economy, which provides some very preliminary insights into what may be possible on a larger scale with "diffuse computing."

7.1 Flexible network monitoring with FLAME

The work discussed in the subsection is due to Anagnostakis, Greenwald, Ioannidis, Miltchev, and M. Smith. [10, 9, 11]. Mr. Anagnostakis is a graduate student at the University of Pennsylvania supported by this project. Related FLAME software documentation is available on this project's web site
`http://www.cis.upenn.edu/spyce/`

The bulk of research on *Active Networks* [182, 176] has been directed towards building general infrastructure [7, 187], with relatively little research driven by particular network functions. Recently, the focus has shifted slightly as researchers seek network functions with a *clear need* for some form of extensibility, and design appropriate *domain-specific* active-network substrates. The main experimental question to be answered in such cases is whether the desired level of flexibility can be achieved while meeting the safety and performance requirements of the particular function in question. Several experiments have been reported in this direction, including the use of smart packets for network management [172] and the design of a system for flexible intra-domain routing[156].

A third such function, that we discuss in this paper, is *network monitoring*, which is becoming increasingly critical for supporting the reactive mechanisms needed to make the Internet more efficient, robust and

secure. Network providers need to analyze properties of network traffic in order to adequately provision and fine-tune the network infrastructure. Furthermore, the network occasionally finds itself in abnormal situations caused by distributed denial of service attacks, network routing outages, *etc.*. Real-time monitoring can potentially detect such conditions and react promptly. Finally, analysis of traffic is needed for network management, accounting and the verification of compliance with diverse policies.

When examining current network monitoring architectures, the weaknesses of the basic abstractions used and the need for a more flexible interface becomes evident. Static implementations of monitoring systems are unable to keep up with evolving demands. The main issue is that routers offer a set of *built-in* monitoring functions but router vendors only implement functions that are cost-effective: those that are interesting to the vast majority of possible customers. If one needs functionality that is not implemented on the router, then it becomes difficult, if not impossible, to extract the needed data from the routers. Furthermore, as customer interests evolve, router vendors can only add functionality on the time-scale of product design and release; it can be months or years from the time customers first indicate interest until a feature makes it into a product. Another critical issue is that the need for timely deployment cannot always be met at the current pace of standardization or software deployment, especially in cases such as detection and prevention of denial-of-service attacks.

The thrust of our research is to determine whether programmable traffic monitoring systems that are flexible enough to be useful, and safe enough to be deployed, can perform well enough to be practical. Here we report on the design and implementation of FLAME, a system for flexible network monitoring. FLAME decouples packet monitoring from packet forwarding, offering a flexible monitoring substrate operating in an environment of common IP forwarders. Our design has three main features. First, it allows efficient implementation of functions that cannot be easily integrated in current router designs. Second, it offers robustness through the use of lightweight protection mechanisms to prevent crashes or information leaks due to malicious or misbehaving functions. Third, it provides a flexible policy model allowing users with different degrees of trust to use the system.

Aspects of our work that were novel at the time have now been validated by their acceptance in the larger community. In network measurement, the advantages of directly using scripts or code operating on packets rather than publishing sanitized traces for offline processing are now widely recognized [141, 154, 53]. Additionally, the Cyclone-based safety model first used in FLAME has been adopted in other systems for the purpose of allowing safe general-purpose kernel extensions [36] and for upgrading TCP algorithms on end-hosts through safe mobile code [157].

Our OpenBSD-based prototype, tested on a 1 GHz Pentium PC, provides approximately 1,300 processing cycles per packet for monitoring modules on a fully-loaded 1 Gbit/s link. A typical workload on our system was measured to consume approximately 800 cycles per packet. The safety overhead for the same workload was measured to be approximately 25%.

7.2 Estimating network internal delays

The work discussed in this subsection is due to Anagnostakis and Greenwald [8]. Related software documentation on network tomography software (cing tool) is available on this project's web page `http://www.cis.upenn.edu/spyce/`.

Network parameters (e.g., delay, loss, and throughput) are relatively easy to measure over an end-to-end path. In contrast, measuring the same quantities on an individual link inside the network is more difficult. The *network tomography* problem requires a remote source to estimate network-internal statistics on individual links within the network based on a series of probes.

Broadly speaking, there are three approaches to estimating the distribution of queuing delays on a particular remote network link. First, one can directly measure the round trip delay to the head and tail of the link (*cf.* [110, 67]). Historically, the first tools that directly measured per-link delays were based on round-trip measurements. Direct measurement tools based on round-trip measurements, such as *pathchar* [110] and similar techniques [67, 130] have many attractions: the simplicity of the direct measurements, coupled with the ability to function without requiring cooperation from remote hosts or routers along some path. This technique requires only the ability to send a packet over a given link, to the tail of the link, without any restrictions on the path taken to get to that link. Two ICMP ping packets are "simultaneously" sent to the tail of the link, but one packet has the TTL set to expire at the head of the link. We subtract the round-trip time of the TTL-expired message from the RTT of the ICMP ECHO reply packet to get an estimate of the instantaneous delay on a link. However, such tools only measure round-trip times and not one-way delays, and asymmetric return paths make it difficult to isolate per-hop estimates (per-hop estimates are only accurate if the return path is the reverse of the forward path).

A second alternative is to indirectly infer internal network statistics by injecting probe packets from one source to multiple destinations and correlating the observed packet behavior on the resulting tree topology [5, 69, 68, 122]. This technique elegantly recovers the delay on remote links as long as we can find pairs of cooperating receivers whose paths directly diverge before and after the link we wish to measure. The first problem with this approach is finding willing receivers in correct locations to isolate the links you are interested in. Second, even if a sufficient number of conveniently placed receivers were found, [13] shows that this indirect technique is not robust (errors in specific cases may be large), accuracy decreases when paths take many hops, and distortions creep in when there are correlations between delays encountered on different links. Further, it can be computationally expensive and its failure mode is to silently give an incorrect result.

The third approach, implemented by tools such as cing [12, 13] and Tulip [126] is based on direct measurement using ICMP Timestamp packets. These are the most accurate of the recent tools. They provide accurate one-way per-hop delay estimates, and use only existing infrastructure. However, cing and Tulip can only determine delays on a link (or multi-hop segment) when routes from the measurement source to *both* ends of that link follow the same path. Internet routes are not generally that regular, and therefore direct measurement techniques based on ICMP Timestamp have limited applicability when applied to specific, individual, links.

We describe a new hybrid approach based on several observations. First, determining delay distributions from indirect inference works quite well if the tree is small. Consequently we infer distributions on a given link from a small number of measured multi-hop segments in the neighborhood of the target. Second, we can use indirect inferencing to distinguish between forward-path and return-path congestion, making RTT measurements practical as input to infer many one-way delays. This means almost any internal node can act as a "cooperating receiver". Third, we can use deconvolution (as described in [14]) to indirectly infer delay distributions even in instances where we do not have measurements of correlated pairs of packets over the target link, and conventional indirect methods are not applicable.

We have built a tool, `cing+`, using these ideas. We find that for any given target link, many different instances of these techniques are possible. In [14]), we describe measurements that rank the accuracy of each approach in different contexts. `cing+` chooses a specific technique for each link after doing a topological analysis of the results of sending TTL-limited probes to each link. `cing+` then returns one-way per-link queuing delay estimates for every hop along the path from the source to the destination. `cing+` provides coverage comparable to or better than indirect inference and round-trip based direct measurements, with accuracy comparable to the one-way direct measurement.

The coverage of this technique is assessed by inspecting many thousands of Internet paths. The results

show that our technique is applicable to almost *any* link in the network. The accuracy of the different probe configurations of `cing+` is evaluated through simulation where knowledge of the real delay distributions is obtainable, and where such real distributions can be compared to estimated delay distributions. Based on our experiments, we show that `cing+` can be applied to many more links than the best previous direct measurement approaches to measure one-way queuing delay, and the measurement techniques used by `cing+` are more accurate than existing indirect inference delay measurement techniques.

7.3 Architecture and performance of server-directed transcoding

The work discussed in this subsection is due to Knutsson, Lu, Mogul, and Hopkins [115]. Dr. Knutsson is a postdoctoral researcher at the University of Pennsylvania supported by this project. Related software documentation on server-directed transcoding is available on this project's web site `http://www.cis.upenn.edu/spyce/`.

Many web site designers face a dilemma: they must balance the richness of the user experience against the limited bandwidth and user interfaces of many Web clients. Media content create particular problems: for example, large images increase download times, and may be hard to display on small screens. The Internet is seeing growth not just in users with wideband access, but also in people using portable narrowband clients with small displays.

To avoid discouraging the latter users, site designers often employ fewer, smaller, or lower-quality images than they would otherwise prefer. Even so, many users encounter sites with excessive image complexity, either because a thoughtful site designer was unwilling to unduly compromise the experience of well-connected users, or because a thoughtless site designer failed to consider the impact of image size.

One can cope with this mismatch between content and capabilities by *transcoding* content to a more appropriate representation. For example, images can be reduced in size, cropped to eliminate details, or converted to monochrome. Transcoding can provide the user with the essential information of the original content, without straining the client or network capabilities.

Transcoding is lossy: while it preserves essential information, it removes ("distills") inessential or unrenderable information, in order to meet goals such as bandwidth reduction. A transcoding system must make a tradeoff between loss of detail and loss of effectiveness at meeting its goals. Too little distillation, and the bandwidth costs (for example) will still be prohibitive; too much distillation, and the underlying message is lost.

Transcoding can be done *a priori* at the origin server, but it is often done at a proxy. Proxy-based transcoding can improve performance and scaling. Traditional transcoding proxies operate autonomously; they make transcoding decisions based on heuristics and implicit information, such as the HTTP "Content-type" header. Implicit information can be ambiguous, so autonomous transcoders can make bad tradeoffs.

This problem inspired one of us to propose *server-directed transcoding*, or SDT, in a previous paper [142]. In this approach, the origin server provides explicit guidance to the transcoding system about whether and how to convert between representations. While autonomous transcoding breaks the end-to-end model of the Web, because the proxy does not know the semantics of the content, SDT preserves end-to-end semantics. SDT also supports more aggressive content transformation than can autonomous transcoding, because there is no risk of accidentally eliminating important content.

Here we show how SDT can be integrated into the HTTP protocol, with a compatible and simple extension. We present the architecture and implementation of a proxy system that supports SDT, using a combination of mobile applets and native code. We discuss several specific, useful transformations for image content. Finally, we report on some simple experiments with our system, including performance

measurements.

Our goal has not been to find a single, optimal solution to the problem of content adaptation. Rather, we studied a pre-existing region of the design space (proxy-based transcoding) to find a solution that respects the end-to-end model, without requiring incompatible changes to existing infrastructure and protocols.

7.4 Peer-to-peer support for massively multiplayer games

The work discussed in this subsection is due to Knutsson, Lu, Xu, and Hopkins [116].

We propose the use of peer-to-peer (P2P) overlays to support massively multi-player games (MMGs) on the Internet. Players participating in the game form an overlay on which many of the game functions are implemented. The players thus contribute the memory, CPU cycles and bandwidth to manage the shared game state.

The premise of most MMGs is that of a large shared game world inhabited by thousands of players. The emphasis is often on social interactions and exciting story lines. Games like Lineage have recorded two million registered players, and 180K concurrent players in one night.

Online MMGs are traditionally supported by a client-server architecture, where the server keeps both player account information and handles game state. Scalability is achieved by employing server clusters. The servers can either be connected by LANs, as in Terazona [190], or they can form a computing grid, as in Butterfly.net [40]. Although this architecture scales with the number of players, it lacks flexibility and the server has to be over-provisioned to handle peak loads. Furthermore, the client-server model limits the deployment of user-designed games, which is an important trend in game design. While games like EverQuest allow limited user designed game extensions, security and performance concerns will limit the scope of such extensions since they would need to be hosted on the game servers handling the core game.

Massively multiplayer online games are natural applications for peer-to-peer overlays. We take advantage of the self-organizing characteristic of P2P overlays to create a system that dynamically scales up and down with the number of players. Game players also have incentives to join the overlay, because the participation is limited to the duration of the player's game play.

Games are different from previous P2P applications that focus on the harnessing of idle storage and network bandwidth, including storage systems [54, 168, 51], content distribution [44, 109] and instant messaging [137]. Games utilize the memory and CPU cycles of peers to maintain the shared game state. Three potential problems must be addressed to make this approach fully applicable in practice:

- **Performance**—games have frequent updates, that must be propagated under certain time constraints. Furthermore, peers have limited bandwidth since they are located at the edge of the network.
- **Availability**—replicating game states to improve availability has two potential problems. First, once a peer goes offline, its state quickly becomes stale and the replica becomes invalid. Secondly, because of high update frequency, maintaining a large set of replicas is a potential performance bottleneck.
- **Security**—both the prevention of account thefts and the prevention of cheating during game play should be considered. Distributing game states to the peers increases the opportunities for cheating.

We address the first two problems in detail. Our design prevents account thefts, the problem most important to the game industry, by centralizing the account management at the server, and only distributing game states to the peers. Although cheat-prevention is a major concern for online games [23], it is a separate issue from the basic performance and availability of P2P gaming. We will make note of instances where

cheat-prevention has influenced our design, but the details and particulars are a subject of ongoing and future work.

The primary technical contributions of this work are architectural and evaluative. We preset a novel architecture marrying massively multiplayer games with peer-to-peer networking technologies, and we provide a detailed performance study to demonstrate the feasibility of our design.

The key to the feasibility of a P2P game architecture is locality of interest [145]. Games are designed such that while the game world is large, the area of interest to a single player is limited, typically correlating to the sensory capabilities of the game characters being modeled. The players, in turn, can be arranged into groups with coinciding areas of interest. This self-organizing property of MMGs matches the self-organizing character of peer-to-peer networks. In particular, nearby (as defined in game terms) players can form peer groups, and keep updates to game state within the group.

References

[1] M. Abadi. On SDSI's linked local name spaces. *Journal of Computer Security*, 6(1-2):3–21, 1998.

[2] Martín Abadi and Andrew D. Gordon. A calculus for cryptographic protocols: the spi calculus. *Information and Computation*, 143:1–70, 1999. Expanded version available as SRC Research Report 149 (January 1998).

[3] Martín Abadi and Phillip Rogaway. Reconciling two views of cryptography (the computational soundness of formal encryption). *Journal of Cryptology*, 15(2):103–127, 2002.

[4] Alessandro Acquisti, Roger Dingledine, and Paul Syverson. On the economics of anonymity. In Rebecca N. Wright, editor, *Financial Cryptography, FC 2003*, volume 2742 of *LNCS*. Springer Verlag, 2003.

[5] A. Adams, T. Bu, R. Caceres, N. Duffield, T. Friedman, J. Horowitz, F. Lo Presti, S. Moon, V. Paxson, and D. Towsley. The use of end-to-end multicast measurements for characterizing internal network behavior. *IEEE Communications Magazine*, 38(5):152–159, May 2000.

[6] W. Aiello, S.M. Bellovin, M. Blaze, R. Canetti, A.D. Keromytis J. Ioannidis, and O. Reingold. Just fast keying (JFK), 2002. Internet draft.

[7] D. Scott Alexander, William A. Arbaugh, Michael W. Hicks, Pankaj Kakkar, Angelos D. Keromytis, Jonathan T. Moore, Carl A. Gunter, Scott M. Nettles, and Jonathan M. Smith. The SwitchWare active network architecture. *IEEE Network*, 12(3):29–36, May/June 1998.

[8] K. G. Anagnostakis and M. B. Greenwald. A hybrid direct-indirect estimator of network internal delays. In *Proc. of ACM SIGMETRICS'04*, 2004.

[9] K. G. Anagnostakis, M. B. Greenwald, S. Ioannidis, and S. Miltchev. Open Packet Monitoring on FLAME: Safety, Performance and Applications. In *Proceedings of the 4th International Working Conference on Active Networks (IWAN'02)*, December 2002.

[10] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, J. Ioannidis, Michael B. Greenwald, and J. M. Smith. Efficient packet monitoring for network management. In *Proceedings of the 8th IFIP/IEEE Network Operations and Management Symposium (NOMS)*, pages 423–436, April 2002. (an earlier extended version of this paper is available as UPenn Technical Report MS-CIS-01-28, September 2001).

[11] K. G. Anagnostakis, S. Ioannidis, S. Miltchev, J. Ioannidis, and J. M. Smith. Efficient packet monitoring for network management. Technical Report MS-CIS-01-28, Computer and Information Science, The University of Pennsylvania, September 2001.

[12] Kostas Anagnostakis, Michael B. Greenwald, and Raphael S. Ryger. cing: Measuring network-internal delays using only existing infrastructure. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, April 2003.

[13] Kostas G. Anagnostakis and Michael B. Greenwald. Direct measurement vs. indirect inference for determining network internal delays. *Performance Evaluation*, 49(1-4):165–176, September 2002. (Performance '02).

[14] Kostas G. Anagnostakis and Michael B. Greenwald. A hybrid approach to estimating per-link network queueing delays. Technical Report MS-CIS-02-31, CIS Department, University of Pennsylvania, December 2003. (supersedes earlier version of October, 2002).

[15] R. Ankney, D. Johnson, and M. Matyas. The unified model, 1995. Contribution to X9F1.

[16] The Anonymizer. `http://www.anonymizer.com/`.

[17] A. Archer, J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Approximation and collusion in multicast cost sharing. *Games and Economic Behavior*, 47:36–71, 2004.

[18] A. Archer and É. Tardos. Frugal path mechanisms. In *Proceedings of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02)*, pages 991–999. ACM Press/SIAM, New York, 2002.

[19] N. Asokan, Matthias Schunter, and Michael Waidner. Optimistic protocols for fair exchange. In *4th ACM Conference on Computer and Communications Security*, Zurich, Switzerland, April 1997. ACM Press.

[20] N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18(4):593–610, 2000.

[21] Michael Backes, Birgit Pfitzmann, and Michael Waidner. Reactively secure signature schemes. In *Proceedings of 6th Information Security Conference*, volume 2851 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 2003.

[22] Michael Backes, Birgit Pfitzmann, and Michael Waidner. A general composition theorem for secure reactive systems. In *Proceedings of 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*. Springer, 2004.

[23] Nathaniel E. Baughman and Brian Neil Levine. Cheat-proof playout for centralized and distributed online games. In *INFOCOM '01*, pages 104–113, 2001.

[24] Birgit Baum-Waidner and Michael Waidner. Round-optimal and abuse free optimistic multi-party contract signing. In *Automata, Languages and Programming — ICALP 2000*, volume 1853 of *Lecture Notes in Computer Science*, pages 524–535, Geneva, Switzerland, July 2000. Springer-Verlag.

[25] D. Beaver. Foundations of secure interactive computing. In *Crypto'91*, volume 576 of *Lecture Notes in Computer Science*, pages 377–391. Springer-Verlag, 1991.

[26] D. Beaver. Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority. *Journal of Cryptology*, 4:75–122, 1991.

[27] G. Bella. *Inductive Verification of Cryptographic Protocols*. PhD thesis, University of Cambridge, March 2000. `<http://www.cl.cam.ac.uk/~gb221/papers/bella14.ps.gz>`.

[28] G. Bella and L. C. Paulson. Using Isabelle to Prove Properties of the Kerberos Authentication System. In H. Orman and C. Meadows, editors, *Proc. of DIMACS'97, Workshop on Design and Formal Verification of Security Protocols (CD-ROM)*, 1997. `<http://www.cl.cam.ac.uk/~gb221/papers/bella4.ps.gz>`.

[29] G. Bella and L. C. Paulson. Kerberos Version IV: Inductive Analysis of the Secrecy Goals. In *Proc. of ESORICS '98, Fifth European Symposium on Research in Computer Science*, number 1485 in Lecture Notes in Computer Science, pages 361–375. Springer-Verlag, 1998. `<http://www.cl.cam.ac.uk/~gb221/papers/bella6.ps.gz>`.

[30] G. Bella and L. C. Paulson. Mechanising BAN Kerberos by the Inductive Method. In *Proc. of CAV98 – Tenth International Conference on Computer Aided Verification*, 1998. `<http://www.cl.cam.ac.uk/~gb221/papers/bella5.ps.gz>`.

[31] G. Bella and E. Riccobene. Formal Analysis of the Kerberos Authentication System. *J. Universal Comp. Sci.*, 3(12):1337–1381, December 1997.

[32] M. Ben-Or, O. Goldreich, S. Micali, and R. Rivest. A fair protocol for signing contracts. *IEEE Transactions on Information Theory*, 36(1):40–46, 1990.

[33] Oliver Berthold, Andreas Pfitzmann, and Ronny Standtke. The disadvantages of free MIX routes and how to overcome them. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*. Springer-Verlag, LNCS 2009, July 2000.

[34] M. Blaze, J. Feigenbaum, J. Ioannidis, and A. Keromytis. The role of trust management in distributed system security. In J. Vitek and C. Jensen, editors, *Secure Internet Programming: Security Issues for Distributed and Mobile Objects*, pages 185–210. Springer LNCS 1603, 1999.

[35] D. Boneh and M. Naor. Timed commitments and applications. In *Proc. CRYPTO 'OO*, pages 236–254, 2000.

[36] Herbert Bos and Bart Samwel. Safe kernel programming in the OKE. In *Proceedings of IEEE OPENARCH 2002*, June 2002.

[37] Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *Proceedings of the Royal Society, Series A*, 426(1871):233–271, 1989. Also appeared as SRC Research Report 39 and, in a shortened form, in ACM Transactions on Computer Systems 8, 1 (February 1990), 18-36.

[38] F. Butler, I. Cervesato, A. D. Jaggard, and A. Scedrov. An Analysis of Some Properties of Kerberos 5 Using MSR. In *Proceedings of the 15th Computer Security Foundations Workshop*, 2002.

[39] Frederick Butler, Iliano Cervesato, Aaron D. Jaggard, and Andre Scedrov. A Formal Analysis of Some Properties of Kerberos 5 Using MSR. Technical Report MS-CIS-04-04, University of Pennsylvania Department of Computer and Information Science, April 2004. Extended version of [38].

[40] Butterfly.net, Inc. The butterfly grid: A distributed platform for online games, 2003. www.butterfly.net/platform/.

[41] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[42] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. 42nd IEEE Symp. on the Foundations of Computer Science*. IEEE, 2001. Full version available at `http://eprint.iacr.org/2000/067/`.

[43] Ran Canetti, Yehuda Lindell, Rafail Ostrovsky, and Amit Sahai. Universally composable two-party and multi-party secure computation. In *Proc. ACM Symp. on the Theory of Computing*, pages 494–503, 2002.

[44] M. Castro, P. Druschel, A-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. Splitstream: High-bandwidth content distribution in a cooperative environment. In *IPTPS'03*, 2003.

[45] R. Chadha, J. C. Mitchell, A. Scedrov, and V. Shmatikov. Contract signing, optimism, and advantage. *Journal of Logic and Algebraic Programming, Special issue on Processes and Security*, 2004. To appear.

[46] Rohit Chadha, Steve Kremer, and Andre Scedrov. Formal analysis of multi-party contract signing. In *17th IEEE Computer Security Foundations Workshop*, pages 266–279, Asilomar, CA, USA, June 2004. IEEE Computer Society Press.

[47] Rohit Chadha, John C. Mitchell, Andre Scedrov, and Vitaly Shmatikov. Contract signing, optimism, and advantage. In *CONCUR 2003 — Concurrency Theory*, volume 2761 of *LNCS*, pages 361–377. Springer-Verlag, 2003.

[48] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[49] Endless BGP convergence problem in Cisco IOS software releases., 2001. Cisco Field Note. `http://www.cisco.com/warp/public/770/fn12942.html`.

[50] S.H. Clearwater, editor. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. World Scientific, 1996.

[51] Clip2. The Gnutella protocol specification v0.4 document revision 1.2, 2000. www9.limewire.com/developer/.

[52] ContentGuard. XrML: The digital rights language for trusted content and services. http://www.xrml.org/, 2001.

[53] Jan Coppens, Steven Van den Berghe, Herbert Bos, Evangelos Markatos, Filip De Turck, Arne Oslebo, and Sven Ubik. SCAMPI: A scalable and programmable architecture for monitoring gigabit networks. In *Proceedings of the Workshop on End-to-End Monitoring Techniques and Services (E2EMON), in conjunction with 6th IFIP/IEEE International Conference on Management of Multimedia Networks and Services (MMNS)*, September 2003.

[54] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of SOSP'01*, October 2001.

[55] George Danezis, Roger Dingledine, and Nick Mathewson. Mixminion: Design of a type III anonymous remailer protocol. In *2003 IEEE Symposium on Security and Privacy*, pages 2–15. IEEE CS, May 2003.

[56] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. A derivation system for security protocols and its logical formalization. In *Proceedings of 16th IEEE Computer Security Foundations Workshop*. IEEE, 2003.

[57] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition. Proceedings of Mathematical Foundations of Programming Semantics, to appear in *Electronic Notes in Theoretical Computer Science*, 2003.

[58] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Secure protocol composition (Extended abstract). In *Proceedings of ACM Workshop on Formal Methods in Security Engineering*, pages 11–23, 2003.

[59] A. Datta, A. Derek, J. C. Mitchell, and D. Pavlovic. Abstraction and refinement in protocol derivation. In *Proceedings of 17th IEEE Computer Security Foundations Workshop*. IEEE, 2004.

[60] Anupam Datta, Ralf Küsters, John C. Mitchell, Ajith Ramanathan, and Vitaly Shmatikov. Unifying equivalence-based definitions of protocol security. In *Proceedings of ACM SIGPLAN and IFIP WG 1.7 4th Workshop on Issues in the Theory of Security*, 2004.

[61] C. Diaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In R. Dingledine and P. Syverson, editors, *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *Lecture Notes in Computer Science*, pages 54–68, Berlin/New York, 2002. Springer-Verlag.

[62] Whitfield Diffie and Susan Landau. *Privacy On the Line: The Politics of Wiretapping and Encryption*. MIT Press, 1998.

[63] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[64] Roger Dingledine, Vitaly Shmatikov, and Paul Syverson. Synchronous batching: From cascades to free routes. In David Martin and Andrei Serjantov, editors, *Privacy Enhancing Technologies, PET 2004*.

[65] D. Dolev and A. Yao. On the security of public-key protocols. *IEEE Transactions on Information Theory*, 2(29), 1983.

[66] Danny Dolev and Andrew C. Yao. On the security of public key protocols. *IEEE Transactions on Information Theory*, 29(2):198–208, 1983.

[67] Allen B. Downey. Using pathchar to estimate Internet link characteristics. In *Proceedings of ACM SIGCOMM*, pages 241–250, September 1999.

[68] N. G. Duffield, J. Horowitz, F. Lo Presti, and D. Towsley. Network delay tomography from end-to-end unicast measurements. In *Proc. of the 2001 International Workshop on Digital Communications 2001 - Evolutionary Trends of the Internet*, September 2001.

[69] Nick G. Duffield and F. Lo Presti. Multicast inference of packet delay variance at interior network links. In *Proceedings of IEEE INFOCOM 2000*, April 2000.

[70] Nancy Durgin, Patrick Lincoln, John Mitchell, and Andre Scedrov. Multiset rewriting and the complexity of bounded security protocols. *Journal of Computer Security*, 12:247–311, 2004.

[71] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[72] E. Elkind, A. Sahai, and K. Steiglitz. Frugality in path auctions. In *Proceedings of the 15th ACM-SIAM Symposium on Discrete Algorithms (SODA '04)*, pages 701–709. ACM Press/SIAM, New York, 2004.

[73] C. Ellison. SPKI Certificate Documentation. http://world.std.com/~cme/html/spki.html.

[74] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Communications of the ACM*, 28(6):637–647, 1985.

[75] S. Even and Y. Yacobi. Relations among public key signature schemes. Technical Report 175, Computer Science Dept. Technion, Israel, March 1980.

[76] J. Feigenbaum, A. Krishnamurthy, R. Sami, and S. Shenker. Hardness results for multicast cost sharing. *Theoretical Computer Science*, 304:215–236, 2003.

[77] J. Feigenbaum, C. Papadimitriou, R. Sami, and S. Shenker. A BGP-based mechanism for lowest-cost routing. In *Proceedings of the 21st ACM Symposium on Principles of Distributed Computing (PODC '02)*, pages 173–182. ACM Press, New York, 2002.

[78] J. Feigenbaum, C. Papadimitriou, and S. Shenker. Sharing the cost of multicast transmissions. *Journal of Computer and System Sciences*, 63:21–41, 2001.

[79] J. Feigenbaum, R. Sami, and S. Shenker. Mechanism design for policy routing. In *Proceedings of the 23rd Symposium on Principles of Distributed Computing (PODC'04)*, pages 11–20. ACM Press, New York, 2004.

[80] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *Proceedings of the 6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communication (DIALM '02)*, pages 1–13. ACM Press, New York, 2002.

[81] M. Fischer, N. Lynch, and M. Patterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[82] M. J. Fischer. The consensus problem in unreliable distributed systems. Technical Report RR-273, Yale University, 1983. Also appears in *Foundations of Computation Theory*, ed. M. Karpinski, Lecture Notes in Computer Science, Vol. 185, Springer Verlag, 1983, pp. 127–140.

[83] T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31:469–472, 1985.

[84] J. Garay, M. Jakobsson, and P. MacKenzie. Abuse-free optimistic contract signing. In *Proc. CRYPTO'99*, pages 449–466, 1999.

[85] Juan A. Garay and Philip D. MacKenzie. Abuse-free multi-party contract signing. In *International Symposium on Distributed Computing*, volume 1693 of *Lecture Notes in Computer Science*, Bratislava, Slovak Republic, September 1999. Springer-Verlag.

[86] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, New Jersey, 2002.

[87] J. Glasgow, G. MacEwen, and P. Panangaden. A logic for reasoning about security. *ACM Transactions on Computer Systems*, 10(3):226–264, 1992.

[88] S. Goel, M. Robson, M. Polte, and E. G. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Unpublished manuscript, 2002.

[89] Oded Goldreich. *The Foundations of Cryptography*, volume 1. Cambridge University Press, June 2001.

[90] Oded Goldreich. *The Foundations of Cryptography*, volume 2. June 2003. Manuscript under preparation; latest version available at `http://www.wisdom.weizmann.ac.il/~oded/foc-vol2.html`.

[91] S. Goldwasser and L. Levin. Fair computation of general functions in presence of immoral majority. In *Crypto'90*, volume 537 of *Lecture Notes in Computer Science*, pages 77–93. Springer-Verlag, 1990.

[92] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In Tatsuaki Okamoto, editor, *Topics in Cryptology – CT-RSA 2004*, volume 2964 of *LNCS*. Springer Verlag, 2004.

[93] D. Gollmann. Authentication–Myths and Misconceptions. *Progress in Computer Science and Applied Logic*, 20:203–225, 2001.

[94] J. W. Gray and P. F. Syverson. A logical approach to multillevel security of probabilistic systems. *Distributed Computing*, 11(2):73–90, 1998.

[95] T. G. Griffin, A. D. Jaggard, and V. Ramachandran. Design Principles of Policy Languages for Path-Vector Protocols. In *Proceedings of ACM SIGCOMM'03*, pages 61–72. ACM Press, New York, 2003. Extended version: Yale Univ. Tech. Report 1250 (YALEU/DCS/TR-1250), Apr. 2004 `ftp://ftp.cs.yale.edu/pub/TR/tr1250.{ps,pdf}`.

[96] T. G. Griffin, F. B. Shepherd, and G. Wilfong. The stable paths problem and interdomain routing. *ACM/IEEE Trans. on Networking*, 10(2):232–243, 2002.

[97] J. Y. Halpern and K. O'Neill. Secrecy in multiagent systems. In *Proc. 15th IEEE Computer Security Foundations Workshop*, 2002.

[98] J. Y. Halpern and K. O'Neill. Anonymity and information hiding in multiagent systems. In *Proc. 16th IEEE Computer Security Foundations Workshop*, pages 75–88, 2003.

[99] J. Y. Halpern and M. R. Tuttle. Knowledge, probability, and adversaries. *Journal of the ACM*, 40(4):917–962, 1993.

[100] J. Y. Halpern and R. van der Meyden. A logic for SDSI's linked local name spaces. In *Proc. 12th IEEE Computer Security Foundations Workshop*, pages 111–122, 1999. To appear, *Journal of Computer Security*.

[101] J. Y. Halpern, R. van der Meyden, and F. Schneider. Logical foundations for trust management. Manuscript, 1999.

[102] Joseph Halpern and Vicky Weissman. Using first-order logic to reason about policies. In *Proceedings of the 16th IEEE Computer Security Foundations Workshop*, pages 187–201, 2003.

[103] Joseph Halpern and Vicky Weissman. A formal foundation for xrml. In *Proceedings of the 17th IEEE Computer Security Foundations Workshop*, pages 251–263, 2004.

[104] Joseph Y. Halpern and Kevin R. O'Neill. Anonymity and information hiding in multiagent systems. *Journal of Computer Security*. To appear.

[105] J. Hershberger and S. Suri. Vickrey prices and shortest paths: What is an edge worth. In *Proceedings of the 42nd Symposium on the Foundations of Computer Science*, pages 129–140. IEEE Computer Society Press, Los Alamitos, 2001.

[106] D. Hughes and V. Shmatikov. Information hiding, anonymity and privacy: a modular approach. *Journal of Computer Security*, 12(1):3–36, 2004.

[107] R. Iannella. ODRL: The open digital rights language initiative. http://odrl.net/, 2001.

[108] ITU-T Rec. X.509 (revised). The Directory - Authentication Framework. International Telecommunication Union, 1993.

[109] Sitaram Iyer, Antony Rowstron, and Peter Druschel. Squirrel: A decentralized peer-to-peer Web cache. In *PODC*, July 2002.

[110] Van Jacobson. pathchar - A tool to infer characteristics of Internet paths. *ftp://ftp.ee.lbl.gov/pathchar*, April 1997.

[111] A. Jaggard and V. Ramachandran. Robustness of class-based path-vector systems. In *Proceedings of the 12th International Conference on Network Protocols*. IEEE Computer Society Press, Los Alamitos, 2004.

[112] A. D. Jaggard and V. Ramachandran. Relating two formal models of path-vector routing. In *Proceedings of IEEE INFOCOM 2005*.

[113] Stanislaw Jarecki, Patrick Lincoln, and Vitaly Shmatikov. Negotiated privacy. In M. Okada, B. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Software Security - Theories and Systems. Mext-NSF-JSPS International Symposium, ISSS 2002*, volume 2609 of *LNCS*, pages 96–111. Springer Verlag, 2003.

[114] Richard A. Kemmerer, Catherine Meadows, and Jonathan K. Millen. Three systems for cryptographic protocol analysis. *Journal of Cryptology*, 7(2):79–130, 1994.

[115] Bjorn Knutsson, Honghui Lu, Jeffrey Mogul, and Bryan Hopkins. Architecture and performance of server-directed transcoding. *ACM Transactions on Internet Technology (TOIT)*, 3(4):392–424, November 2003.

[116] Bjorn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games. In *Proc. of IEEE INFOCOM 2004*, Hong Kong, China, March 2004.

[117] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5), September 1993. Network Working Group Request for Comments: 1510. `<ftp://ftp.isi.edu/in-notes/rfc1510.txt>`.

[118] H. Krawczyk. Sigma: The sign-and-mac approach to authenticated diffie-hellman and its use in the IKE protocols. In *Advances in Cryptology - CRYPTO 2003*, volume 2729, pages 400–425. Springer-Verlag Heidelberg, 2003.

[119] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. Technical Report 98-05, COOR, 1998.

[120] B.N. Levine and C. Shields. Hordes: A multicast based protocol for anonymity. *Journal of Computer Security*, 10(3):213–240, 2002.

[121] N. Li. *Delegation Logic: A Logic-based Approach to Distributed Authorization*. PhD thesis, New York University, September 2000.

[122] Gang Liang and Bin Yu. Pseudo likelihood estimation in network tomography. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies (Infocom 2003)*, March 30 - April 3 2003.

[123] Patrick D. Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. A probabilistic poly-time framework for protocol analysis. In Michael K. Reiter, editor, *Proc. 5th ACM Conference on Computer and Communications Security*, pages 112–121, San Francisco, California, 1998. ACM Press.

[124] Patrick D. Lincoln, John C. Mitchell, Mark Mitchell, and Andre Scedrov. Probabilistic polynomial-time equivalence and security protocols. In Jeannette M. Wing, Jim Woodcock, and Jim Davies, editors, *Formal Methods World Congress, vol. I*, number 1708 in Lecture Notes in Computer Science, pages 776–793, Toulouse, France, 1999. Springer.

[125] G. Lowe. Some new attacks upon security protocols. In *Proc. 9th IEEE Computer Security Foundations Workshop*, pages 162–169, 1996.

[126] Ratul Mahajan, Neil Spring, David Wetherall, and Thomas Anderson. User-level Internet path diagnosis. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 106–119, Oct. 19–22 2003.

[127] O. Markowitch and S. Saeednia. Optimistic fair exchange with transparent signature recovery. In *Proc. 5th International Conference on FinancialCryptography*, pages 339–350, 2001.

[128] A. Mas-Collel, W. Whiston, and J. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[129] Paulo Mateus, John C. Mitchell, and Andre Scedrov. Composition of cryptographic protocols in a probabilistic polynomial-time process calculus. In Roberto M. Amadio and Denis Lugiez, editors, *14th International Conference on Concurrency Theory*, volume 2761 of *Lecture Notes in Computer Science*, pages 327–349, Marseille, France, 2003. Springer-Verlag.

[130] M. Mathis and J. Mahdavi. Diagnosing Internet congestion with a transport-layer performance tool. In *Proceedings of INET'96*, June 1996.

[131] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *The Transactions of the IECE of Japan*, E69:99–106, 1986.

[132] D. McPherson, V. Gill, D. Walton, and A. Retana. BGP persistent route oscillation condition, 2002. RFC 3345.

[133] C. Meadows. A cost-based framework for analysis of denial of service in networks. Journal of Computer Security, to appear. Available in preprint form at http://chacs.nrl.navy.mil/publications/CHACS/CRYPTOindex.html.

[134] Catherine Meadows. Analyzing the Needham-Schroeder public-key protocol: A comparison of two approaches. In *Proc. European Symposium On Research In Computer Security*, pages 351–364. Springer Verlag, 1996.

[135] Catherine Meadows. The NRL protocol analyzer: An overview. *Journal of Logic Programming*, 26(2):113–131, February 1996.

[136] S. Micali and P. Rogaway. Secure computation. In *Crypto'91*, volume 576 of *Lecture Notes in Computer Science*, pages 392–404. Springer-Verlag, 1991.

[137] Alan Mislove, Ansley Post, Charles Reis, Paul Willmann, Peter Druschel, Dan S. Wallach, Xavier Bonnaire, Pierre Sens, Jean-Michel Busca, and Luciana Arantes-Bezerra. Post: A secure, resilient, cooperative messaging system. In *9th Workshop on Hot Topics in Operating Systems (HotOS IX)*, may 2003.

[138] J.C. Mitchell and V. Teague. Autonomous nodes and distributed mechanisms. In M. Okada, B. Pierce, A. Scedrov, H. Tokuda, and A. Yonezawa, editors, *Software Security - Theories and Systems. Mext-NSF-JSPS International Symposium, ISSS 2002*, volume 2609 of *LNCS*, pages 58–83. Springer Verlag, 2003.

[139] John C. Mitchell, Mark Mitchell, and Ulrich Stern. Automated analysis of cryptographic protocols using Murφ. In *Proc. IEEE Symposium on Security and Privacy*, pages 141–151, 1997.

[140] John C. Mitchell, Ajith Ramanathan, Andre Scedrov, and Vanessa Teague. A probabilistic polynomial-time calculus for the analysis of cryptographic protocols (preliminary report). In Stephen Brookes and Michael Mislove, editors, *17th Annual Conference on the Mathematical Foundations of Programming Semantics, Arhus, Denmark, May, 2001*, volume 45. Electronic notes in Theoretical Computer Science, 2001.

[141] J. Mogul. Trace anonymization misses the point. Presentation on WWW 2002 Panel on Web Measurements.

[142] Jeffrey C. Mogul. Server-directed transcoding. In *Proc. 5th International Web Caching and Content Delivery Workshop*, Lisbon, Portugal, May 2000.

[143] Ulf Möller, Lance Cottrell, Peter Palfrader, and Len Sassaman. Mixmaster Protocol — Version 2. Draft, July 2003. `http://www.abditum.com/mixmaster-spec.txt`.

[144] D. Monderer and M. Tennenholtz. Distributed games: from mechanisms to protocols, 2000.

[145] Katherine L. Morse. Interest management in large-scale distributed simulations. Technical Report ICS-TR-96-27, University of California, Irvine, 1996.

[146] MPEG. MPEG-21 rights expression language FCD. http://www.chiariglione.org/mpeg/working_documents.htm, 2003.

[147] R.M. Needham and M.D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

[148] C. Neuman, J. Kohl, T. Ts'o, Ken Raeburn, and Tom Yu. The Kerberos Network Authentication Service (V5), November 20 2001. Internet draft, expires 20 May 2002. <`http://www.ietf.org/internet-drafts/draft-ietf-cat-kerberos-revisions-10.txt`>.

[149] C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, September 1994.

[150] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5), February 15 2004. Internet draft, expires 15 August 2004. <`http://www.ietf.org/internet-drafts/draft-ietf-krb-wg-kerberos-clarifications-05.txt`>.

[151] Noam Nisan and Amir Ronen. Algorithmic mechanism design. *Games and Economic Behavior*, 35:166–196, 2001.

[152] M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

[153] H. Pagnia and F. Gaertner. On the impossibility of fair exchange without a trusted third party. Technical Report TUD-BS-1999-02, Department of Computer Science, Darmstadt University of Technology, Germany, March 1999.

[154] Ruoming Pang and Vern Paxson. A high-level programming environment for packet trace anonymization and transformation. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM)*, pages 339–351. ACM Press, 2003.

[155] D. Parkes. iBundle: An efficient ascending price bundle auction. In *Proc. ACM Conference on Electronic Commerce (EC-99)*, 1999.

[156] Craig Partridge, Alex Snoeren, Tim Strayer, Beverly Schwartz, Matthew Condell, and Isidro Castineyra. FIRE: Flexible intra-AS routing environment. In *Proceedings of ACM SIGCOMM*, pages 191–203. August 2000.

[157] Parveen Patel, Andrew Whitaker, David Wetherall, Jay Lepreau, and Tim Stack. Upgrading transport protocols using untrusted mobile code. In *Proceedings of the 19th ACM Symposium on Operating systems Principles (SOSP)*, pages 1–14. ACM Press, 2003.

[158] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th IEEE Computer Security Foundations Workshop*, pages 70–83, Washington - Brussels - Tokyo, June 1997. IEEE.

[159] S. Payette and T. Staples. The Mellon Fedora project: Digital library architecture meets xml and web services. In *European Conference on Research and Advanced Technology for Digital Libraries*, pages 406–421, 2002. http://www.fedora.info/documents/ecdl2002final.pdf.

[160] A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudeonymity: a proposal for terminology. In *International Workshop on Designing Privacy Enhancing Technologies*, pages 1–9, New York, 2001. Springer-Verlag.

[161] Birgit Pfitzmann and Michael Waidner. Composition and integrity preservation of secure reactive systems. In *7th ACM Conference on Computer and Communications Security*, pages 245–254, Athens, November 2000. ACM Press. Preliminary version: IBM Research Report RZ 3234 (# 93280) 06/12/00, IBM Research Division, Zürich, June 2000.

[162] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical report tr-81, Harvard Aiken Computation Laboratory, 1981.

[163] Ajith Ramanathan, John C. Mitchell, Andre Scedrov, and Vanessa Teague. Probabilistic bisimulation and equivalence for security analysis of network protocols. In Igor Walukiewicz, editor, *7th International Conference, FOSSACS*, volume 2987 of *Lecture Notes in Computer Science*, pages 468–483, Barcelona, Spain, 2004. Springer-Verlag.

[164] I.B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *Journal of Cryptology*, 8(4):201–222, 1995.

[165] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[166] R.L. Rivest and B. Lampson. Cryptography and Information Security Group Research Project: A Simple Distributed Security Infrastructure. www.toc.lcs.mit.edu/˜cis/sdsi.html.

[167] A. W. Roscoe. Modelling and verifying key-exchange protocols using CSP and FDR. In *8th IEEE Computer Security Foundations Workshop*, pages 98–107. IEEE Computer Society, 1995.

[168] Antony Rowstron and Peter Druschel. Storage management and caching in PAST, A large-scale, persistent peer-to-peer storage utility. In Greg Ganger, editor, *Proceedings of SOSP-01*, volume 35, 5 of *ACM SIGOPS Operating Systems Review*, pages 188–201, New York, October 21–24 2001. ACM Press.

[169] Peter Ryan, Steve A. Schneider, Michael H. Goldsmith, Gavin Lowe, and A. W. Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2001.

[170] Steve A. Schneider. Verifying authentication protocols with CSP. In Simon Foley, editor, *10th IEEE Computer Security Foundations Workshop*, pages 3–17, Rockport, Massachusetts, USA, June 1997. IEEE Computer Society Press.

[171] Steve A. Schneider. Formal analysis of a non-repudiation protocol. In *11th IEEE Computer Security Foundations Workshop*, pages 54–65, Washington - Brussels - Tokyo, June 1998. IEEE.

[172] B. Schwartz, A. Jackson, T. Strayer, W. Zhou, R. Rockwell, and C. Partridge. Smart packets: Applying active networks to network management. *ACM Transactions on Computer Systems*, 18(1):67–88, February 2000.

[173] A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In R. Dingledine and P. Syverson, editors, *Proc. Privacy Enhancing Technologies Workshop (PET 2002)*, volume 2482 of *Lecture Notes in Computer Science*, pages 41–53, Berlin/New York, 2002. Springer-Verlag.

[174] Andrei Serjantov, Roger Dingledine, and Paul Syverson. From a trickle to a flood: Active attacks on several mix types. In Fabien Petitcolas, editor, *Information Hiding (IH 2002)*. Springer-Verlag, LNCS 2578, 2002.

[175] R. Sherwood, B. Bhattacharjee, and A. Srinivasan. P5: A protocol for scalable anonymous communication. In *IEEE Symposium on Security and Privacy*, pages 58–70, 2002.

[176] J. M. Smith, K. Calvert, S. Murphy, H. Orman, and L. Peterson. Activating networks: a progress report. *IEEE Computer*, 32(4), April 1999.

[177] J. L. Sobrinho. Network routing with path vector protocols: Theory and applications. In *Proceedings of ACM SIGCOMM'03*, pages 49–60, 2003.

[178] I.E. Sutherland. A futures market in computer time. *Communications of the ACM*, 11:449–451, 1968.

[179] P. F. Syverson, D. M. Goldschlag, and M. G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, 1997.

[180] P. F. Syverson and S. G. Stubblebine. Group principals and the formalization of anonymity. In *World Congress on Formal Methods*, pages 814–833, 1999.

[181] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *SPIE ITCom 2001*, pages 188–195. SPIE Press, Bellingham, August 2001.

[182] D.L. Tennenhouse, J.M. Smith, W.D. Sincoskie, D.J. Wetherall, and G.J. Minden. A survey of active network research. *IEEE Communications Magazine*, pages 80 – 86, January 1997.

[183] K. Varadhan, R. Govindan, and D. Estrin. Persistent route oscillations in inter-domain routing. *Computer Networks*, 32:1–16, 2000.

[184] L. von Ahn, A. Bortz, and N. J. Hopper. k-anonymous message transmission. In *10th ACM Conference on Computer and Communications Security*, pages 122–130, 2003.

[185] C.A. Waldspurger, T. Hogg, B.A. Huberman, J.O. Kephart, and W.S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions on Software Engineering*, 18:103–117, 1992.

[186] M. P. Wellman. A market-oriented programming environment and its applications to distributed multicommodity flow problems. *Journal of AI Research*, 1:1–23, 1993.

[187] David Wetherall. Active network vision and reality: Lessons from a capsule-based system. In *Proceedings of the 17th ACM Symposium on Operating System Principles (SOSP)*, pages 64 – 79, December 1999.

[188] Andrew C.-C. Yao. Theory and applications of trapdoor functions. In *IEEE Foundations of Computer Science*, pages 80–91, 1982.

[189] J. Zhou and D. Gollmann. A fair non-repudiation protocol. In *Proc. IEEE Symposium on Security and Privacy*, pages 55–61, 1996.

[190] Zona Inc. Terazona: Zona application frame work white paper, 2002. www.zona.net/whitepaper/Zonawhitepaper.pdf.