DAISY: Increasing Scalability and Robustness of Anonymity Systems

Chi-Bun Chan and Cristina Nita-Rotaru
Department of Computer Sciences
Purdue University
West Lafayette, IN 47906, USA
{cbchan,crisn}@cs.purdue.edu

***Abstract*— Common anonymity system designs fall into two categories. Some use a more reliable architecture relying on a fixed and relatively small set of core relays, while other use a dynamic peer-to-peer (P2P) infrastructure to draw an open-ended pool of relays and provide increased scalability. Both approaches have limitations. The first design has limited scalability and allows an adversary to focus on the few entry and exit points to infer traffic correlations, while the second design can be problematic in keeping the overall system stable and operating due to unpredictable user behavior and system complexity.**

In this paper we propose a hybrid architecture for anonymity systems with the consideration of scalability, robustness and quality-of-service, by taking the advantages from both static and P2P designs as well as balancing the incurred tradeoffs. We describe the design in details and discuss its potential benefits.

I. Introduction

A variety of cryptographic mechanisms has been developed and deployed to achieve basic security objectives such as confidentiality, integrity and authentication. In addition to these basic services, there is an increasing concern on privacy and anonymity. There is a potential market for providing anonymizing services and related products. For example, anonymous web-surfing helps users to protect their privacy against censorship of their online behavior and communications. Anonymizing mechanisms can be used to hide the existence of tunneled connections and prevent particular servers from being identified as attack targets.

Common requirements of anonymity protection include *sender anonymity*, *receiver anonymity* and *unlinkability*. An anonymizing service preserves sender anonymity if, given a set of users and a particular message or communication, one cannot achieve a higher probability of identifying the sender (initiator) of that message after the service is applied. Similarly, receiver anonymity concerns on whether the recipient of a particular message is identified. Unlinkability, or more precisely, *relationship anonymity* means that the correspondence of a sender and receiver pair is not revealed.

In this paper, we refer to the class of systems which provide anonymizing services as *anonymity systems*[1], and the associated underlying networks as *anonymizing networks*.

For some applications, like web-browsing and instant messages, long response time and network delays are commonly considered annoying and unacceptable. In order to accommodate the low latency requirement, anonymity systems face the difficulties in defending against effective attacks like timing analysis. Even if the systems switch to high-latency approaches, they are still vulnerable to some stronger attacks like long-term intersection [1] and disclosure attacks [2]–[4].

Many mix-based systems (e.g. [5], [6]) and Onion Routing [7] have a fixed and relatively small set of core relays and the initiator and responder of communications are usually distinguishable from relays. This makes an adversary easily identify the traffic initiator or responder at the endpoints provided he controls the first or the last relay in the route. Scalability also becomes a concern in these static designs when the users and traffic volume grow. Dynamic peer-to-peer (P2P) based systems like Tarzan [8] and MorphMix [9], on the other hand, have the potential for drawing an open-ended pool of relays and solving the scalability problem. However, with user-operated nodes as a part of the system, unpredictable user behavior can be problematic in keeping the overall system stable and easily maintained.

In this paper we propose a hybrid architecture for anonymity systems which provides increased scalability and robustness, by taking advantage of both static and P2P designs, as well as balancing the incurred tradeoffs. Such an architecture offers the flexibility to separate concerns, distribute responsibilities and adopt different tradeoff strategies among the sub-components in the architecture. More specifically, our contribution consists of:

- We provide a brief survey and classification of major attacks on anonymity systems and position our work in this context.
- We present a hybrid architecture that integrates principles drawn from static and P2P designs. A key component of proposed architecture is extending the idea of anonymous circuits and introduce the concept of delegates to partition the circuits across the sub-systems in our hybrid architecture. More specifically, we break the path into three sub-paths: submitting path, bridging path and collecting path, each with its own anonymizing and routing mechanisms.
- We discuss the potential benefits of the proposed architecture. We believe that such an architecture offers increased scalability, fault-tolerance and stronger anonymizing services as each sub-components can adopt different anonymizing and routing strategies.

[1]These systems are also referred as anonymous communication systems or anonymity-providing systems

The rest of this paper is organized as follows. Section II overviews some anonymity system designs and their mechanisms. Section III summarizes the main attacks against anonymity systems. Section IV states the assumptions and goals of our proposed system. We present our system in Sections V and VI and provide a discussion in Section VII. Finally, Section VIII concludes our work.

II. Related Work

Many designs of anonymity systems have use relay nodes to anonymize traffic. Anonymizing systems can be classified as message-based and connection-based [10]. The former include mix-based systems which follow the concept introduced by Chaum [11]. The latter, also referred as low latency designs, aim at providing anonymity protection on interactive traffic.

Mix-based systems utilize one or more intermediate network nodes to relay messages, and, at the same time, break the correspondence among incoming and outgoing messages of a relay by changing their appearance and temporal information. This is referred as the basic batch-and-mix operation, in which a mix collects a batch of fixed-length messages from different sources, cryptographically transforms the batched messages and then forwards a subset of messages to their recipients in a random order. A variety of mix-based designs have later been developed based on different topologies, message batching and release policies, and route selection strategies [12]–[17].

Low latency designs rely on the concept of *anonymous circuit* in which several relays are used to forward messages in a multi-hop fashion. Some systems, like Crowds [18], implement the anonymous circuit by extending the route of a message to a random node probabilistically. By doing so, every predecessor sending traffic to a node would have certain probability to be either the initiator or just a forwarder, and thus obfuscate the real initiator. Other systems, like Onion Routing [7], [19], try to hide the route of messages by using fixed-length layer-encrypted structures to embed traffic.

With respect to the topology, some designs like Tarzan [8] and MorphMix [9] follow the idea of anonymous circuits, but explore the P2P architecture for the underlying anonymizing network, instead of using a fixed set of core relays as in Onion Routing based systems. As every peer utilizes the anonymizing services and at the same time operates as a relay, the overall anonymizing network has a potentially larger and more dynamic set of relays. However, the P2P nature of these anonymizing networks induces a much higher risk of facing colluding malicious nodes controlled by adversaries. Some designs thus provide collusion prevention (e.g. peer selection protocol in Tarzan) and collusion detection (e.g. detection strategy in MorphMix) mechanisms to address this issue.

Incentive is another important issue in P2P designs. The authors in [20] studied the economics in anonymity systems and pointed out the incentive problem as a major barrier to wide deployment of decentralized anonymity infrastructures. Several incentive mechanisms [21]–[25] were proposed to encourage better cooperation and reduce free-riding in P2P systems, ranging from global economic and reputation to local exchange- or reward-based models.

Another type of anonymity systems employs anonymous broadcast or multicast. Chaum proposed the Dining Cryptographer protocol [26] which provides strong anonymity with information-theoretic proofs. Later designs such as P^5 [27] and Herbivore [28] evolved into P2P architecture with hierarchically organized topology for the sake of efficiency and scalability. For the hierarchical approaches, the overall system still uses a single architecture. Unlike the previous systems, our work utilizes a hybrid architecture approach in designing an anonymity system with separate security concerns and tradeoffs across the underlying sub-systems that can adopt different anonymizing and routing strategies.

III. Attacks in Anonymity Systems

In this section we provide a brief summary and classification of the adversary's power and goals as well of the common attacks in anonymity systems. Our classification does not mean to be exhaustive and we recommend interested readers to refer to [1], [2], [4], [10], [29]–[37] for details.

A. Adversary's Power

An adversary can be *passive* or *active*. A passive adversary only eavesdrops traffic while an active adversary can observe, modify, inject and drop messages in the anonymity system.

The capability of adversary observing and manipulating traffic can be *local* or *global*. The former can only access traffic within one or only very few autonomous networks while the latter is capable to observe effectively a large amount or all network links within the system. The global observability is a much stronger assumption on the adversarial power, which also imposes a greater challenge on designing a secure anonymity system. This strong assumption is reasonable for anonymizing networks with a fixed and small set of core nodes, particularly for those operated by one or a few parties. For anonymizing networks with a potentially larger and distributed set of participating nodes, however, it is less likely that an adversary can capture the global view of traffic. This is usually true in P2P approaches, which feature not only a larger but more dynamic set of intermediate nodes.

An adversary can also be viewed as *internal (insider)* or *external (outsider)*. An insider participates in the system as a normal user or operator by running his own nodes but acts dishonestly or maliciously. An outsider, in contrast, is generally out of the system's scope and without the privilege of operating the anonymizing services. It can, however, access the communication links in the anonymizing network.

B. Adversary's Goals

1) Degrading the quality of the anonymizing services: A common goal of an adversary attacking anonymity systems is to break the anonymizing service or at least degrade the quality of anonymity protection. An adversary would try to break the sender anonymity by linking messages or communications to their originator. Similarly the receiver anonymity can be broken if he can figure out the corresponding message recipient. The adversary can further exploit the unlinkability by revealing sender and receiver correspondence of messages.

Basic cryptographic tools, like encryption and message authentication code, cannot guarantee anonymity protection. Communications in an encrypted channel could be vulnerable to several traffic analysis and traffic confirmation attacks.

2) Decreasing the utilization of the anonymity system: User experience and satisfaction affect the utilization of a system. Poor performance, frequent down-time and annoying experience may outweigh security from the user's perspective, particularly in non-critical applications. By decreasing the performance, reliability and availability of an anonymity system, an attacker may successfully drive users switching to a lower protection level or completely not using the targeted system. The potential threat of this kind of attacks can be severe and thus should not be underestimated. The attacker could achieve similar effects in degrading the quality of anonymizing service but without spending the effort on traffic analysis. Denial-of-Service (DoS) attacks are one major type of attacks that render anonymity systems unresponsive and unavailable.

C. Attack Description

Many attacks in anonymity systems were also discussed in [1], [2], [4], [10], [29]–[37]. Based on the characteristics of the attacks, we see some commonalities among them which can be classified as *traffic-specific* and *system-specific*. Some attacks that exploit both areas can fit into both categories.

Traffic-specific attacks are usually based on observation and inference. Many anonymity systems achieve anonymity by obfuscating the correlation between input and output messages. In order to break the anonymity, one has to track messages within the anonymizing network or at the endpoints. *Message feature* and *traffic dynamics* are two major sources of hints that ease the analysis. Through passively eavesdropping or actively manipulating traffic within the anonymizing network, an adversary can observe any distinguishable message features and traffic patterns. By exploiting correlations between traffic and nodes, the adversary may be able to infer the route, the sender, the receiver and the sender-receiver correspondence of a given message or communication. End-to-end traffic confirmation, the disclosure attack [32] and statistical disclosure attacks [2]–[4] are examples of attacks in this category.

System-specific attacks break anonymity protection based on the design, limitations and flaws of a given anonymity system. For example, several systems implement link padding or traffic shaping policy; an adversary may use these properties to isolate a particular message and to trace its route. Resource exhaustion like flooding or DoS attacks can saturate network links, cause some nodes unable to operate and weaken the whole anonymity system. Selfishness attack, which is specific to P2P designs, also creates bad impacts on the system.

1) Attacks Based on Message Feature: Message feature characterizes the static attributes of the traffic. It is a basic source of leaking information that can expose identification of the sender and receiver. Variable message sizes may provide hints to help adversary distinguish between different types of messages and track the message routes.

2) Attacks Based on Traffic Dynamics: Traffic dynamics captures the dynamic behavior of messages over a communication session, including packet count, message volume, message frequency, timing information, communication patterns and intersections of active sender-recipient groups at different times. Through enough observations, an adversary may find correlations between incoming messages and outgoing messages across nodes in the anonymizing network and further deduce message routes and sender-receiver mappings.

3) End-to-End Traffic Confirmation: Observing the endpoints in an anonymity system is one simple way to track the sender-receiver correspondence. The intermediate structure of the anonymity systems can be simply abstracted as a black-box, independent of how messages are routed, transformed or mixed within the given systems. By eavesdropping the traffic passing through two endpoints on a suspected route, an adversary can study the correlations between messages entering and leaving the anonymous tunnel. The goal of the adversary is to identify to which successor of the exit node the traffic from a particular predecessor of the entry node is sent. If the entry and the exit node happen to be the first and the last node respectively in an anonymous tunnel, the adversary can figure out the sender-receiver pair of a communication.

If an anonymity system does not handle traffic ”carefully”, some traffic patterns and characteristics may be exposed, helping the adversary to infer the correlations. An eavesdropper can count the number of messages that enter and exit the two endpoints or can measure the inter-message timings and message frequencies. A timing attack can be mounted to correlate the timings of a message at the entry node with those coming out of the exit node. This information may help the adversary to map some input message to output messages, and rule out potential senders or receivers from the anonymity sets. Through operating or compromising nodes in the system, an active adversary can drop, delay or mark messages to increase the effectiveness in exploiting the correlations.

4) Disclosure Attack and Statistical Disclosure Attacks (mainly target for high-latency designs):

a) The Disclosure Attack: Kesdogan, Agrawal and Penz [32] described a traffic analysis technique, called disclosure attack, which is used to infer all possible recipients of a targeted sender in an anonymity system. The attack is a repetitive inference process which consists of a learning phase and an excluding phase. It is based on the observations of receiver anonymity sets over the time.

The adversary assumes the number of possible recipients of a targeted sender s to be m. During the learning phase, he collects, as the basis sets, m mutually disjoint recipient sets such that each of them contains exactly one recipient of s, by observing the incoming and outgoing messages of s.

The attack then proceeds to the excluding phase in which the adversary collects additional recipient sets and prunes out from each basis set those entries that are not corresponding recipients of s. For every later recipient set R' he observes, if R' is mutually disjoint from all but one basis set, he can shrink the non-disjoint basis set to its intersection with R'. He repeats the analysis by refining the value of m until he successfully deduces the possible recipient set of s.

b) Statistical Disclosure Attacks: Danezis [2] pointed out that obtaining an optimal solution in the disclosure attack is NP-hard. He proposed an approximation method called statistical disclosure attack. In this attack, the adversary observes the receiver anonymity sets corresponding to the messages sent by a targeted sender s. He gathers a sequence of observation vectors $\vec{o_i}$ where each vector represents the probability distribution of each recipient being the recipient of the message sent by s at a particular time. He also obtains the probability distribution $\vec{u}$ of the background traffic that is not related to s. By collecting a large enough set of $\vec{o_i}$, the adversary can estimate the recipient set of s from $\vec{u}$ and the arithmetic mean of $\vec{o_i}$ by the Law of Large Numbers. The idea was further extended to more realistic mix-based systems in [3], [4].

5) Resource Exhaustion and Denial-of-Service Attacks: Pre-set system constraints, limitations on network link bandwidth, node computational power and available system resources generally exist in practical implementations of anonymity systems. These constraints may allow an attacker to launch various resource exhaustion attacks. A flooding-based DoS attack is one common example. By saturating some communication links or overwhelming some processing components (nodes, routers, etc), a portion of the network may exhibit notable changes in the ongoing traffic and several nodes may behave differently. Deliberately dropped packets or selectively forwarding messages can be considered as another kind of DoS attacks which can also render identifiable traffic changes. These attacks would allow an adversary to perform further traffic analysis more efficiently. For instance, an attacker can count any packet dropping or perform timing analysis to measure any increased latency on suspected routes. By paralyzing part of the network or flooding the network with identifiable traffic, an adversary may also isolate a targeted message from others so as to trace the route of that message.

6) Selfishness Attack: In P2P anonymizing networks, a participant is expected to serve other peers while it receives services from others. Unfortunately, some nodes may want to free-ride the services without contributing their resources. For instance, some selfish nodes may selectively forward messages or ignore incoming requests from its neighbors. Their behaviors appear similarly as the second type of DoS attacks mentioned before, but due to their selfishness rather than malicious intentions. Although they do not intentionally attack the system, they can weaken the systems and facilitate attacks coming from other adversaries.

IV. Assumptions and System Goals

A. Assumptions

Our anonymity system, DAISY[2] (hybriD architecture for AnonymIty SYstems), is overlaid on top of the IP infrastructure. Users can join the system and use the anonymizing network for communication. We assume the underlying IP network is insecure such that an attacker may observe, inject or spoof IP packets without being detected. Our anonymity system does not protect against IP spoofing, but this issue can be addressed by using secure protocols like IPSEC [38].

As a part of our protocols, we assume a public-key infrastructure (PKI) is supported. This infrastructure can be a Certificate Authority (CA)-based, where a distributed cluster of peer CAs sharing a common certificate and revocation list can be deployed to improve the CA's availability.

B. System Goals

Regarding to the anonymizing services, we aim to provide protections with three different areas of concerns, in term of message, anonymity and privacy respectively.

Message protection covers confidentiality which allows disclosure of messages to designated recipients, and integrity which resists message modification during transmission.

Anonymity protection is the core service which provides anonymizing mechanisms for IP-level point-to-point communications. The ultimate goal is to preserve, to a certain degree, sender anonymity, receiver anonymity and unlinkability with respect to traffic and communication under different circumstances. By "certain" we mean the anonymity protection should withstand some proposed traffic analysis attacks, however, it cannot guarantee a prefect anonymity solution. More specifically, we try to prevent attacks based on message feature, frustrate attacks based on traffic dynamics by either providing preventive measures or making the attacks much harder in practice. For the latency requirement, we offer the flexibility in the system to accommodate low latency and high latency services with the tradeoff in the anonymizing power.

Privacy protection refers to filtering privacy-sensitive contents in messages. As we mentioned in section III, a message itself may leak out information of the communicating parties and thus the sender should handle messages properly before transmitting them via the anonymizing network. However, privacy filtering is protocol and application dependent. We can certainly pay special attention to common protocols like HTTP, FTP and DNS. Yet it is very challenging to come up with a generic and exhaustive filtering engine as there are always evolving protocols and newly developed applications. In our design, privacy filtering is not our main focus and we only provide basic filtering functions such as address rewriting. Nevertheless, users can use specific privacy filtering tools like Privoxy [39] on top of DAISY to enhance privacy protection.

From the system perspective, we want to achieve a certain degree of fault tolerance on the underlying infrastructure regarding the situations due to heavy traffic load, unexpected errors, malicious behavior and attacks, peer instability and network dynamics.

V. System Architecture

Below we describe the motivations behind our design and provide a high-level description of our system.

A. Motivation

To give a clearer vision of our system, especially the intuitions behind, we highlight in this section the important observations which lead to the foundations of our design. Readers may find in later sections some traits in our system coherent with the intuitions described below.

[2]The name "DAISY" has a three-fold meaning: (1) our architecture shown in Figure 1 resembles a daisy flower; (2) white daisy is a symbol of innocence which coincides with the nature of anonymity systems which provide certain degree of innocence to users; (3) anonymous circuit is similar to a daisy chain.

First, we observe that the adversarial power varies with situations. In practice, the capabilities of an adversary are mostly constrained by the environments he deals with. Taking the observability as an example again, an adversary could be able to monitor the global traffic within a small and static network, whereas he is less likely to achieve this capability in a largely distributed and dynamic network. Here it becomes a challenge to define policies that realistically capture the capabilities of a possible adversary and satisfy the security objectives. Too loose or too conservative estimation on the adversarial power either harms the security or degrades the usability and performance. The mutual challenge appears when we implement solutions to enforce the policies. This leads to our second observation that the strategies for tackling adversaries can be tailored for special situations. Investing on a general, all-purpose and comprehensive strategy can be very costly, and even worse, it incurs performance and usability tradeoffs in order to handle a broad class of attacks. As an alternative, using a mix of less comprehensive but fine-grained strategies, with each tailored for a specific class of environments and attacks, offers more flexibility in balancing tradeoffs according to application needs.

An anonymity system operated by one or a few organizations is usually a static system with a fixed number of nodes. Scalability becomes a problem under the sustainable growth of users and traffic volume. However, nodes in the system are more reliable and accountable on their behavior, in the sense that, it is unlikely that they intentionally do not follow the protocols or denying their responsibilities. Adversaries to such a system are more likely to be external and they would monitor the network links and even try to get internal access of the system by compromising some nodes.

In contrast, an anonymity system using a P2P architecture has the potential for wide deployment which helps solving the scalability problems. By using nodes from public to operate the system, it provides the benefit of a large, dynamic, distributed and open-ended pool of relays which makes certain attacks harder in practice. The nodes in the P2P system, they are considered unreliable and potentially malicious from a user's point of view. One should not rely on a particular node for the service but he can always choose other peers to pair with. This make the system as a whole more resilient to some corrupted nodes. Regarding the entire P2P system, it is less likely to be under full control by a "Big Brother", provided there are good incentives to attract good users.

There is a tradeoff among security, performance and cost. A hybrid architecture often offers the flexibility to separate concerns, distribute responsibilities and adopt different tradeoff strategies among the sub-components in the architecture. With this in mind, we propose to integrate a more reliable but static system with a more scalable but dynamic system.

B. A High Level Overview

The major goal of DAISY is to support bidirectional anonymous point-to-point communications by which we can attain the anonymity protection mentioned in section IV.

Our design uses the anonymous circuit mechanism as the foundation. However, instead of using one end-to-end circuit connecting the sender and the receiver as endpoints, we break it into three sub-paths by introducing a special kind of intermediates called *delegates*. A delegate acts as the exit point of a circuit that handles any incoming and outgoing messages on behalf of the initiator at the other end of the circuit, which we also refer it as the *master* of the delegate. To visualize it in a simpler way, we can consider there is a circuit connecting the sender s and its delegate d_s. We refer to it as the *submitting path*. The other circuit connecting the receiver r and its delegate d_r forms the *collecting path*, and the routing between the two delegates d_s and d_r forms the *bridging path*. Once the nodes have established anonymous circuits for the submitting and the collecting path, the message transportation then becomes multi-hop forwarding on the three sub-paths.

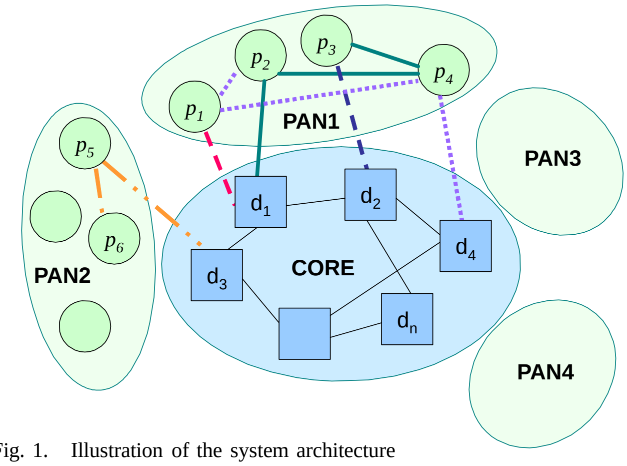

Fig. 1. Illustration of the system architecture

DAISY combines two types of network: *peripheral anonymizing network (PAN)* and *core delegate network (CORE)*. Figure 1 presents an example.

PAN is a decentralized unstructured P2P overlay network. Users desiring to use the anonymizing service will join the PAN and operate as peers using their machines. We will refer to the nodes in PAN as *peers*. Due to the unforeseen nature of peers, PAN is generally composed of untrusted heterogeneous systems with significant variations in computational resources and link bandwidth. There can be multiple PANs in the anonymity system since a peer does not necessarily know the global peer list. PAN covers the submitting and the collecting paths. Anonymous circuits are built within PAN using the participating peers as relays. These circuits terminate at corresponding delegates residing in CORE.

CORE consists of a relatively small static set of nodes that operate as *delegates*. It covers the bridging path and can be viewed as a message exchange network that bridges the submitting and collecting paths. As opposed to multiple PANs, there is only one CORE in the anonymity system.

As all transit messages from PAN will be aggregated into CORE, CORE requires a higher network capacity and computational power to handle the unanticipated amount of traffic. It is thus expected to be operated by parties with more sophisticated machines and networks that provide reliable services. These parties can be organizations which provide free and voluntary services, or companies that charge the services.

The underlying topology and routing mechanism in CORE can vary according to different requirements for the degree of anonymity. In general, the decision of the underlying design considers the balance among security, performance and cost. In other words, the CORE as a whole provides a unified interface with PANs that allow users to specify the security level and utilize the anonymizing services. Every low-level detail about the subsystems should be transparent to users.

VI. Protocol Details

In this paper, unless otherwise specified, we will stick to the notations in Table I in describing the protocols of DAISY.

TABLE I

Notations in the protocol context

$m_1 \parallel m_2$	Concatenation of message m_1 and m_2
d_x	Delegate of x
$d_{j,i}$	The ith delegate in group G_j
T_i	Tag for identifying different types of message, i is the type identification number
PrK_x	Private key of x
PuK_x	Public key of x
$Cert_x$	Signed Certificate of x which includes PuK_x
$E(m,k)$	Encryption of message m with key k (encryption method depends on key type)
$S(m,k)$	Digital signature of message m with public key k
$H(m)$	Cryptographic hashing of message m
$H(m,k)$	Cryptographic keyed-hashing of message m with key k
$EK_{x,y}$	Symmetric encryption key for the link between x and y
$IK_{x,y}$	Integrity key for the link between x and y
SK_i	Segment key associated with the ith relay r_i w.r.t. the given circuit
ID_i	Segment identifier for the ith segment (associated with the ith relay r_i) w.r.t. the given circuit

A. Joining the Network

To join the anonymizing network in DAISY, a node has to connect with some peers in a PAN. This process consists of two phases: peer discovery and neighbor formation.

The peer discovery is bootstrapped via a few information servers or *registries* from which a node can know other active peers. These registries, which may appear as web pages, allow active nodes to publish their information such as IP addresses and public keys. The registries refresh the peer lists to expire any inactive peers. Peers are responsible for contacting the registries regularly and receiving updated information.

When a node wants to join a PAN, it obtains an initial list of active peers from a registry, and registers its information with the registry. The node may also consult several registries to reduce the chance of getting biased information from a malicious registry. Based on the list, a node can select some of the active peers to form a neighborhood. Once it connects with several peers, they can exchange peer information with each other, without bothering the registries. We refer to a pair of peers with a direct link in PAN as *neighbors*.

During the pairing of two peers, they also negotiate two keys associated with their link: one encryption key and one integrity key. The encryption key is used for symmetric encryption (per link basis) to provide confidentiality to sensitive data in transit on the link. The integrity key is used for keyed-hashing to protect message integrity, prevent impersonation and prevent an attacker from injecting messages.

B. End-to-end Message Handling

In end-to-end message handling, privacy-sensitive contents are filtered such that the message itself does not leak out any information about the communicating parties. In addition, end-to-end message confidentiality and integrity are protected. These treatments prevent against attacks on message feature and message tampering. Ideally, every handled message appears as "random text" and does not reveal any information.

The end-to-end message handling is a companion of the anonymous transportation mechanism and they are used together to achieve the anonymous communication. In our design, these two processes are separate and independent of each other. Users can certainly send and receive messages via the anonymous transportation without doing end-to-end message handling but this makes several attacks possible.

Suppose m is the original message that s wants to send to r. The general process of the end-to-end message handling is described as follows:

1. Address re-writing and (optional) privacy filtering ($F(m)$ is the combined filtering function) $q' = F(m)$
2. End-to-end encryption $q'' = E(q', EK_{s,r})$
3. End-to-end integrity check $q = q'' \parallel H(q'', IK_{s,r})$

q is the post-processed message that will be transmitted via the anonymizing network.

C. Transportation via Anonymous Circuit

An anonymous circuit (*circuit* for short) is a multi-hop path established among several peers in PAN and terminating at a delegate in CORE. We use a sequence of nodes as the notation of a circuit. For instance, the kth circuit of peer x with length l is denoted by $C_x^k = x, r_1, r_2, \ldots, r_{l-1}, d_x^k$ where r_i's are the intermediate relays and d_x^k is the corresponding delegate. For the sake of simplicity in expressions, we also refer to x as r_0 and d_x^k as r_l. r_{i-1} and r_i (except r_l) are neighbors in PAN and there is a direct link between them. In the context of a circuit, the link between r_{i-1} and r_i is referred as the ith *segment*, r_{i-1} is the *predecessor* of r_i and r_i is the *successor* of r_{i-1}.

As mentioned before, each pair of neighbors share one encryption key and one integrity key for the associated link. With respect to a circuit, there is one more encryption key per segment basis, which we refer to it as the *segment key*. As several circuits may share a link for one segment, each pair of neighbors could have multiple segment keys. A locally unique *segment identifier* ($segid$) is used to identify a segment between two peers. It is attached on messages in transmitting through a circuit. Each relay has a routing table that contains mappings in form of $(segid_{local}) \rightarrow (successor, segid_{successor})$. Based on the segment identifier of an incoming message, a relay can determine the next relay and the next segment identifier to forward the message.

Given that q is the post-processed message after end-to-end message handling. Consider a peer x is sending q through one of its established anonymous circuits, $C_x^k = x, r_1, r_2, \ldots, d_x^k$. Suppose SK_i is the segment key shared between x and r_i. ID_i is the identifier for the ith segment of the circuit. Before transmitting q down to the circuit, x attaches a *routing tag* to q. The tag, with a format $tag = method \parallel rv$, basically indicates how q should be routed in CORE so as to reach the receiver delegate. The $method$ field specifies the routing method and rv provides the information on how to reach the receiver delegate. The detailed usage of routing tag will be described in section VI-F. The tag is removed by the receiver delegate once the message is on the collecting path.

Afterward, x encrypts the consolidated message repeatedly using the segment keys in reverse order:

$$e_l = E(tag \parallel q, SK_l)$$
$$e_i = E(e_{i+1}, SK_i) \qquad 1 \le i < l$$

e_1 is the final nested encryption of m. x encrypts the identifier of the first segment with EK_{x,r_1} and attaches it to e_1. The whole message is then hashed using IK_{x,r_1}. x finally forwards the consolidated message to r_1. In general, r_{i-1} forwards e''_i to r_i where

$$e'_i = E(ID_i, EK_{r_{i-1},r_i}) \parallel e_i$$
$$e''_i = e'_i \parallel H(e'_i, IK_{r_{i-1},r_i})$$

Upon receiving e''_i, r_i first validates the message integrity. It then decrypts the segment identifier using the link encryption key shared with r_{i-1}. Based on ID_i, r_i can uniquely identify the segment and determine the next relay r_{i+1} on the circuit. It gets e_{i+1} by decrypting e_i with appropriate segment key. It then rewrites and encrypts the next segment identifier, and forwards the consolidated message to r_{i+1}. The process continues until the message finally reaches the delegate d^k_x.

The transportation in reverse direction, i.e. from a delegate to its master, is performed similarly. Each relay decrypts the segment identifier of the incoming message and determines the next relay. It encrypts the message with appropriate segment key, rewrites and encrypts the next segment identifier, and forwards the updated message to the next relay. The nested encryption of the original message will finally reach the master. The master can then repeatedly unwrap the nested encryption using the segment keys in order.

The nested encryption obscures the appearance of messages which frustrates those attacks based on message feature. An adversary cannot directly correlate the incoming and outgoing through a relay based on message contents. The link-based encryption also hides the circuit (segment identifiers) taken by a message and the integrity check prevents segment identifier from altered. The nested encryption together with end-to-end message handling protects against message tagging attacks.

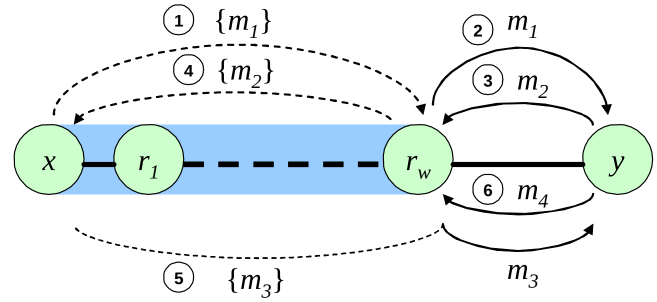

Fig. 2. Extending a partial circuit during circuit establishment

TABLE II

MESSAGES IN THE CIRCUIT ESTABLISHMENT PROTOCOL

$$m_1 = T_{10} \parallel nonce_1$$
$$m_2 = nonce_1 \parallel nonce_2 \parallel IP_y \parallel Cert_y$$
$$m_3 = T_{11} \parallel IP_y \parallel E(nonce_1 \parallel nonce_2 \parallel SK_{w+1}, PuK_y)$$
$$m_4 = T_{12} \parallel ID_{w+1}$$

D. Circuit Establishment and Delegate Association

1) Establishing a Circuit:: We describe the circuit establishment in an inductive fashion. Given a peer x is establishing a new circuit to a delegate. Suppose an incomplete circuit with length w has been established and x is about to extend it one relay further. The process (illustrated in Figure 2) is performed as follows (protocol messages are shown in Table II):

1. x initiates a *circuit extension request* m_1. The request is encapsulated and transmitted as a normal message through the partial circuit. As described in the previous subsection, each relay on the partial circuit unwraps one layer of the nested encryption, rewrites the segment identifier, re-encrypts and forwards the message to the next relay. The request finally appears in clear text at r_w.
2. r_w initiates the request on behalf of x. r_w randomly chooses a neighbor y as the next relay. It forwards the request to y in clear. y either accepts or rejects the request based on its availability to establish further segments.
3. Suppose y accepts the request, it includes in m_2 its certificate issued by a public registry and sends m_2 to r_w.
4. r_w sends m_2 back to x through the partial circuit as if it is a normal message. x on receiving m_2 verifies the reply. If the enclosed certificate is valid, x randomly generates a new segment key SK_{w+1} and returns it as m_3 in an encrypted form using the given public key of y. m_3 is sent to r_w through the partial circuit.
5. On receiving m_3, r_w knows that the initiator agrees y as the next relay r_{w+1} and so it forwards m_3 to y.
6. y returns, as m_4, a locally unique segment identifier ID_{w+1} to r_w such that r_w and y can later identify that segment uniquely by (y, ID_{w+1}) and ID_{w+1} respectively. Both r_w and y update their routing table to include the new segment. y also marks the segment as not terminated.

In case of any problem during the process, an error message is sent to the initiator or to the last relay on the partial circuit.

In the initial case when x wants to establish the first segment, the procedures are the same as what we have mentioned except x now takes also the role of r_w and all the message transportation within the partial circuit can be ignored.

2) Associating a delegate: Once the initiator is satisfied with the length of a circuit, it can complete the circuit by associating a delegate. The initiator sends a *circuit completion request* to the last relay through the partial circuit. The initiator is responsible for selecting a delegate randomly and specifying it on the request. To avoid uneven or biased distribution of associations to some delegates, there can be a centralized server in CORE for assigning delegates upon requests. For simplicity, we leave it as an optional extension to our design.

On receiving the request, the last relay establishes a segment with the chosen delegate in the same way as intermediate relays except an expiration time is returned in addition to a segment identifier by the chosen delegate. The expiration time indicates the lifetime of the circuit and is forwarded to the initiator via the circuit.

3) Security analysis: The circuit is anonymous in the sense that the circuit formation process introduces uncertainty about the actual initiator. By looking at a circuit extension request, a peer generally cannot tell with certainty whether the predecessor is the initiator or just a relay on the circuit (except when the circuit extends back to the initiator and forms a cycle, the initiator certainly notices itself). Unless there exists a global eavesdropper monitoring all traffic or sufficient colluding peers in deducing the origin of the traffic, the initiator can plausibly deny its role by arguing the request is forwarded from its predecessor. The nonces are used in the protocol to assure the freshness of messages and prevent replay attacks.

A malicious relay r_w sitting at the end of the partial circuit can simulate the remaining circuit by replying self-generated public keys. To prevent this last relay from cheating, we can use the witness approach proposed in MorphMix [9].

Consider the reply message m_4 from x, r_w cannot obtain the segment key SK_{w+1} as it is encrypted using y's public key. However, r_w can substitute it with another key. This creates an inconsistency in the segment key shared by x and y. The content of later transmitted messages are messed up as y will wrongly decrypt the nested encryption. Note that r_w cannot play the trick by first decrypting the message with SK_{w+1} and re-encrypting it with the substituted key. So doing the substitution does not benefit r_w much.

In our current design, the initiator chooses the circuit length. An analytical method to optimize the circuit length was proposed in [40]. Although the model considers the optimization globally and closed-form solutions are only allowed in some special cases, a similar approach can be used to derive an estimation of "good" circuit length based on local information of the initiator (with shared information from neighbors).

E. Circuit Switching and Teardown

A delegate association can be terminated when the master leaves the network, or when the association expires. When a peer x leaves the network, it has to tear down all the established circuits. x sends the notification of circuit teardown to associated delegates using the corresponding circuits.

When an association expires, the master peer takes the initiative to notify its delegate. When the notification messages propagate along those circuits, every node on the circuits will update its routing table to tear down the paths.

F. Message Routing in CORE

When a message is forwarded from the sender to a delegate through the submitting circuit, the sender's delegate d_s has to route the message to the receiver's delegate d_r for further delivery. This message routing takes place in CORE.

1) A basic solution: In the simplest case, d_s can directly forward messages to d_r. As one delegate may serve more than one master and have several established circuits, a message arriving at d_r needs to contain information about on which circuit it should be forwarded. One way to achieve this is to use the segment identifiers in d_r's local table since each segment identifier corresponds to a unique circuit. However, this approach certainly leaks out some information about the sender and receiver correspondence. Not only d_s and d_r can identify each other as the involved delegates, but anyone eavesdropping the link between d_s and d_r also knows the fact. Knowing the delegates of sender and receiver does not immediately reveal the actual sender and receiver because the submitting and collecting circuits are still not fully uncovered.

We use the routing tag mentioned in Section VI-C to specify rv as d_r in case of direct forwarding.

2) Anonymous routing on a ring topology: On a ring topology, messages are *circulated* hop-by-hop among the nodes on the ring. Anonymity can be preserved in several ways. First, noise (cover) traffic padding the links on the ring can be used to prevent the packet counting attack. If every pair of consecutive nodes shares a secret key, messages can be encrypted hop-by-hop such that an external observer cannot easily correlate the input and output messages of a node. Rather than specifying the receiver's information on a message, the sender s can put some secret information on the message such that only the intended receiver r can understand. When the message is circulated through r, r can realize the message is destined for it. r can still forward the message to the next hop without revealing that it is the recipient. The sender anonymity can also be protected as a node generally cannot be sure if its predecessor is the sender or just a forwarder unless nodes collude.

3) Anonymous routing in CORE: To fully develop our hybrid solution, we define a second level anonymity protection in CORE. CORE can be viewed as a complete and independent anonymity system in which sender delegates and receiver delegates act like normal senders and receivers, respectively. For efficient message routing, the topology of CORE is expected to have a strong connectivity or exhibit certain hierarchy. For instance, we may connect a collection of ring-based networks in a hierarchical structure. Then the message routing can be performed as intra-ring anonymous routing and inter-ring forwarding. Indeed, we also can implement an anonymous routing scheme based on some proposed designs, like anonymous broadcast style systems [27], [28] or Onion Routing systems [19].

4) Quality-of-service issues: Delegates may adopt different routing methods. This feature does not only serve as an anonymization mechanism but also provides a notion of quality-of-service (QoS). We can derive a simple QoS scheme for DAISY using the routing tag. For each message a sender submits to its delegate, the sender indicates on the routing tag which routing method the sender delegate should use.

Consider a sender delegate d_s is going to forward a message to a receiver delegate d_r. The following shows a possible scheme supporting three QoS levels. The tag for each level is shown inside the parentheses.

(i) Direct forwarding ($tag = direct \parallel d_r$):
d_r is explicitly stated on the tag and d_s directly forwards the message to d_r. This method achieves the lowest latency but the least anonymity protection.

(ii) Anonymous routing with an explicit receiver delegate ($tag = explicit \parallel d_r$):

d_r is explicitly stated on the tag and d_s routes the message anonymously to d_r. This method provides anonymity on d_s but not d_r. The latency varies and depends on the underlying anonymous routing mechanism.

(iii) Anonymous routing with an implicit receiver delegate ($tag = implicit \parallel rv(d_r)$):
d_r is not explicitly known to d_s, but $rv(d_r)$ implicitly tells how to reach d_r. For example, if CORE is implemented as a collection of ring-based networks, $rv(d_r)$ may specify the ring R where d_r is located along with a secret value known to d_s and d_r only. So d_s can forward the message to ring R and then let it circulate on R until d_r catches it based on the secret value.

5) Hardening the CORE: Since CORE is the heart of message exchange in DAISY, the failure of CORE may result in instability, unavailability or complete shutdown of the whole system. The higher network bandwidth and computational power requirement in CORE also poses a burden on maintaining the stability. Fault tolerant is thus a key issue to address. The delegates in CORE can be built from a number of clusters which provides high aggregated network bandwidth and processing power. Cryptographic operations on delegates can also be accelerated by using cryptographic hardware.

G. Exception Handling

1) Breakdown of anonymous circuit: Apart from the normal teardown, a circuit can break down in some exceptional cases. When a peer x on some circuits leaves the network, it breaks the associated circuits. x has to send a breakdown notification to the master and delegate on both sides of any affected circuit. When a delegate fails to handle the delegation, it initiates the notification of circuit breakdown to all associated masters. The notification is sent and handled in the same way as in case of circuit teardown but in reverse direction.

2) Broken anonymous circuit: In case a peer ”dies” or leaves the network unexpectedly, it may cause the ungraceful breakdown of some circuits. When later messages are forwarded along those broken paths, the peers near the break points have to notify the senders. However, message losses may not be avoided. The initiator of a circuit can also send regular heart-beat messages to test the circuit connectivity.

VII. Discussion

A. Security

In the view of security and fault-tolerance, we believe a hybrid architecture can offer several benefits.

a) Mutual protection: Different types of adversaries and attacks exist in PAN and CORE , and thus they are designed accordingly to their threat models. PAN and CORE can be viewed as independent and mostly complete anonymity systems that exhibit different characteristics in structure and anonymizing strategies. They as a whole divide the responsibilities and provide a mutual protection to each other. If the anonymity protection is defeated in either PAN or CORE, the remaining one should still provide enough guarantees to preserve sender anonymity, receiver anonymity or unlinkability.

When the initiator of a circuit in PAN is identified, sender anonymity or receiver anonymity cannot be preserved. In such case, we rely on CORE to protect the unlinkability of sender-receiver correspondence. From the perspective of PAN, CORE operates as an opaque blackbox that obscures the correlation between input and output messages. Without enough power to access traffic inside CORE, an adversary is restricted to infer traffic patterns among different circuits in PANs.

An external global adversary in CORE can monitor all the messages flowing among the delegates and further analyze the communication patterns. With additional colluding delegates, the adversary has a higher chance to resolve particular message routes. The worst case happens when the entire CORE is malicious. In such case, the route of a message within the CORE can be easily traced, and the submitting and collecting circuits partially identified. However, as the nodes in PANs are supposed not to be controlled by CORE, the submitting and collecting circuits still provide protection for the actual sender and receiver. To totally break the anonymity, the malicious parties controlling the CORE have to analyze the links in PANs or collude with the peers.

b) System maintenance: Another benefit appears in maintenance. Even well developed systems require subsequent updates to integrate additional functionalities or fix existing flaws. Maintenance in distributed systems is considered more important as a vulnerability in one participant may cause damages to the entire system. However, it is also more difficult as users may not have enough incentive to keep their software up-to-date and it is possible for the updates to be out-of-sync. It is always better to reduce inconvenience and annoyance to users. As found nowadays in many operating systems, anti-virus and personal firewall utilities, automatic update function at end-user system is thus necessary. But that still does not give a promise until we can strictly enforce the actions. As a compromise, unless for substantial amendments in the whole architecture, system upgrades and security patches can be applied more easily to CORE in a controllable manner. This is also true for upgrading the hardware and network facilities in CORE to meet the unforeseen requirements.

B. Usability

For applications like web-browsing, the servers generally do not participate in the anonymization protocols. To bridge the gap, some peers in PANs can act as as special proxies which forward requests to the actual servers. These nodes can be peers provided by the same parties who operate CORE or other dynamic peers who volunteer for the proxy service.

Security services should also consider human factors. As we mentioned earlier, user satisfaction may have a higher priority over security concerns. Hence, the anonymization and latency requirements may vary with applications, and even within a single application, the requirements may change from time to time. For example, a user may want to temporarily scarify a certain degree of anonymity protection to gain better latency when there is a substantial drop in data rate. Allowing the specification of application requirements complicates the design of an anonymity system. It becomes more subtle to accommodate various application needs in the system while keeping the underlying mechanisms transparent to services. In DAISY, we choose to offer users (particularly senders) the flexibility to control the length of the anonymizing path in order to achieve a balance of tradeoffs.

A user who initiates an anonymity circuit can control the circuit length during the circuit establishment to a delegate. The user (when being a sender) generally cannot control the length of collecting paths, however. In special cases like web-browsing, the peers serving as proxies can agree to limit the collecting paths to a few hops. In low-latency anonymizing services, a very few intermediates are expected. Our architecture at one extreme can be configured to attain anonymization in lower latency by simply short-circuiting the bridging paths. In other words, a sender delegate directly forwards messages to a receiver delegate without doing any anonymous routing in CORE. The leftover anonymizing power relies solely on the anonymity circuits in PANs. In DAISY, a sender can specify to its associated delegate the routing method used in CORE for transporting a particular message. One should note that low-latency and high-latency anonymizing services can coexist in our architecture without any reconfigurations.

C. Scalability

The CORE appears as a bottleneck which may affect the scalability of the overall system. Nevertheless, the nature of CORE facilitates a centralized management on itself which allows system upgrade and scaling to be handled more gracefully. The capacity of CORE can be scaled up in a manageable way for the foreseen expansion of users and traffic. As previously mentioned, we also expect CORE to be a more dedicated infrastructure.

VIII. Conclusions and Future Work

In this paper, we introduced a hybrid architecture approach to address some common issues in static and P2P anonymity systems. Our architecture consists of two main components: PAN network (inspired by P2P systems) and CORE network (inspired by static systems). The essence of PAN is to cope with the dynamic nature of users and turn it into the anonymizing power. CORE can be viewed as the traffic aggregation point and thus the volume and rate of traffic becomes the main concern in CORE. The nature of CORE offers the possibility to achieve a stronger anonymity at the expense of higher computation and communication cost. The segregation of concerns in the two architectures leaves out the space to implement QoS control in CORE. QoS in anonymity systems has not been well discussed but we believe it will be an important issue when anonymizing services become highly demanded. In addition, such a separation can theoretically lead to increased anonymity, since each sub-component can use its own tailored anonymous routing mechanism and achieve better performance and effectiveness.

Several design issues still remain for future work. They include: exploring an efficient mechanism to deal with collusions based on the hybrid nature of our system; extending our design to support anonymous group communications; exploring a framework for CORE to allow users to adjust the trade-off between the degree of anonymity and the performance of communication. Finally, while this paper focuses on the architectural design issues, we are undertaking the prototype implementation and several simulations and experiments to gain insights into the performance and scalability of the proposed architecture.

References

[1] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free MIX routes and how to overcome them," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 30–45.

[2] G. Danezis, "Statistical disclosure attacks: Traffic confirmation in open environments," in *Proceedings of Security and Privacy in the Age of Uncertainty, (SEC2003)*, Gritzalis, Vimercati, Samarati, and Katsikas, Eds., IFIP TC11. Athens: Kluwer, May 2003, pp. 421–426.

[3] G. Danezis and A. Serjantov, "Statistical disclosure or intersection attacks on anonymity systems," in *Proceedings of 6th Information Hiding Workshop (IH 2004)*, ser. LNCS, Toronto, forthcoming.

[4] N. Mathewson and R. Dingledine, "Practical traffic analysis: Extending and resisting statistical disclosure," (Submitted), forthcoming.

[5] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[6] M. Rennhard, S. Rafaeli, L. Mathy, B. Plattner, and D. Hutchison, "An architecture for an anonymity network," in *Proceedings of the IEEE 10th Intl. Workshops on Enabling Technologies (WET ICE 2001)*, Cambridge, MA, USA, June 2001, pp. 165–170.

[7] D. M. Goldschlag, M. G. Reed, and P. F. Syverson, "Hiding Routing Information," in *Proceedings of Information Hiding: First International Workshop*, R. Anderson, Ed. Springer-Verlag, LNCS 1174, May 1996, pp. 137–150.

[8] M. J. Freedman and R. Morris, "Tarzan: A peer-to-peer anonymizing network layer," in *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.

[9] M. Rennhard and B. Plattner, "Practical anonymity for the masses with morphmix," in *Proceedings of Financial Cryptography (FC '04)*, A. Juels, Ed. Springer-Verlag, LNCS (forthcoming), February forthcoming.

[10] A. Serjantov and P. Sewell, "Passive attack analysis for connection-based anonymity systems," in *Proceedings of ESORICS 2003*, October 2003.

[11] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, February 1981.

[12] U. Möller, L. Cottrell, P. Palfrader, and L. Sassaman, "Mixmaster Protocol — Version 2," Draft, July 2003.

[13] G. Danezis, "Mix-networks with restricted routes," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, R. Dingledine, Ed. Springer-Verlag, LNCS 2760, March 2003.

[14] R. Dingledine, V. Shmatikov, and P. Syverson, "Synchronous batching: From cascades to free routes," (Submitted), forthcoming.

[15] D. Kesdogan, J. Egner, and R. Büschkes, "Stop-and-go MIXes: Providing probabilistic anonymity in an open system," in *Proceedings of Information Hiding Workshop (IH 1998)*. Springer-Verlag, LNCS 1525, 1998.

[16] A. Serjantov and R. E. Newman, "On the anonymity of timed pool mixes," in *Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems*. Athens, Greece: Kluwer, May 2003, pp. 427–434.

[17] C. Diaz and A. Serjantov, "Generalising mixes," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*, R. Dingledine, Ed. Springer-Verlag, LNCS 2760, March 2003.

[18] M. Reiter and A. Rubin, "Crowds: Anonymity for web transactions," *ACM Transactions on Information and System Security*, vol. 1, no. 1, June 1998.

[19] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, forthcoming.

[20] A. Acquisti, R. Dingledine, and P. Syverson, “On the Economics of Anonymity,” in *Proceedings of Financial Cryptography (FC ’03)*, R. N. Wright, Ed. Springer-Verlag, LNCS 2742, January 2003.

[21] A. Z. Industries. MojoNation. [Online]. Available: http://www.mojonation.com/

[22] R. Dingledine, N. Mathewson, and P. Syverson, “Reputation in P2P Anonymity Systems,” in *Proceedings of Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[23] Q. Sun and H. Garcia-Molina, “SLIC: A selfish link-based incentive mechanism for unstructured peer-to-peer networks,” in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS’04)*, Hachioji, Tokyo, Japan, March 2004, pp. 506–515.

[24] K. G. Anagnostakis and M. B. Greenwald, “Exchange-based incentive mechanisms for peer-to-peer file sharing,” in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS’04)*, Hachioji, Tokyo, Japan, March 2004, pp. 524–533.

[25] R. T. B. Ma, S. C. M. Lee, J. C. S. Lui, and D. K. Y. Yau, “An incentive mechanism for p2p networks,” in *Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS’04)*, Hachioji, Tokyo, Japan, March 2004, pp. 516–523.

[26] D. Chaum, “The dining cryptographers problem: Unconditional sender and recipient untraceability,” *Journal of Cryptology*, vol. 1, pp. 65–75, 1988.

[27] R. Sherwood, B. Bhattacharjee, and A. Srinivasan, “P5: A protocol for scalable anonymous communication,” in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, May 2002.

[28] S. Goel, M. Robson, M. Polte, and E. G. Sirer, “Herbivore: A Scalable and Efficient Protocol for Anonymous Communication,” Cornell University, Ithaca, NY, Tech. Rep. 2003-1890, February 2003.

[29] J.-F. Raymond, “Traffic Analysis: Protocols, Attacks, Design Issues, and Open Problems,” in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 10–29.

[30] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, “Statistical identification of encrypted web browsing traffic,” in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, Berkeley, California, May 2002, p. 19.

[31] A. Back, U. Möller, and A. Stiglic, “Traffic analysis attacks and tradeoffs in anonymity providing systems,” in *Proceedings of Information Hiding Workshop (IH 2001)*, I. S. Moskowitz, Ed. Springer-Verlag, LNCS 2137, April 2001, pp. 245–257.

[32] D. Kesdogan, D. Agrawal, and S. Penz, “Limits of anonymity in open environments,” in *Proceedings of Information Hiding Workshop (IH 2002)*, F. Petitcolas, Ed. Springer-Verlag, LNCS 2578, October 2002.

[33] B. Pfitzmann and A. Pfitzmann, “How to break the direct RSA-implementation of MIXes,” in *Proceedings of EUROCRYPT 1989*. Springer-Verlag, LNCS 434, 1990.

[34] RProcess, “Selective denial of service attacks,” Usenet post, September 1999.

[35] J. Douceur, “The Sybil Attack,” in *Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002)*, March 2002.

[36] A. Serjantov, R. Dingledine, and P. Syverson, “From a trickle to a flood: Active attacks on several mix types,” in *Proceedings of Information Hiding Workshop (IH 2002)*, F. Petitcolas, Ed. Springer-Verlag, LNCS 2578, October 2002.

[37] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, “Timing attacks in low-latency mix-based systems,” in *Proceedings of Financial Cryptography (FC ’04)*, A. Juels, Ed. Springer-Verlag, LNCS (forthcoming), February forthcoming.

[38] S. Kent and R. Atkinson, “Rfc 2401: Security architecture for the internet protocol,” November 1998.

[39] Privoxy. [Online]. Available: http://www.privoxy.org/

[40] Y. Guan, X. Fu, R. Bettati, and W. Zhao, “An optimal strategy for anonymous communication protocols,” in *Proceedings of the 22nd Intl. Conference on Distributed Computing Systems (ICDCS’02)*, Vienna, Austria, July 2002.