Cashmere: Resilient Anonymous Routing

Li Zhuang, Feng Zhou
U. C. Berkeley
{zl,zf}@cs.berkeley.edu

Ben Y. Zhao
U. C. Santa Barbara
ravenben@cs.ucsb.edu

Antony Rowstron
Microsoft Research UK
antr@microsoft.com

Abstract

Anonymous routing protects user communication from identification by third-party observers. Existing anonymous routing layers utilize Chaum-Mixes for anonymity by relaying traffic through relay nodes called mixes. The source defines a static forwarding path through which traffic is relayed to the destination. The resulting path is fragile and shortlived: failure of one mix in the path breaks the forwarding path and results in data loss and jitter before a new path is constructed. In this paper, we propose Cashmere, a resilient anonymous routing layer built on a structured peer-to-peer overlay. Instead of single-node mixes, Cashmere selects regions in the overlay namespace as mixes. Any node in a region can act as the MIX, drastically reducing the probability of a mix failure. We analyze Cashmere's anonymity and measure its performance through simulation and measurements, and show that it maintains high anonymity while providing orders of magnitude improvement in resilience to network dynamics and node failures.

1 Introduction

In many applications it is desirable to hide the identity of the communicating parties from each other and third-party observers. The ability to anonymously route packets is used in many applications, such as anonymous web browsing [1], anonymous voting and in peer-to-peer applications wanting to ensure fair resource sharing [19].

The first-generation of applications that used anonymous routing, including the Anonymizer [1], were centralized, with central points of failure. More recent anonymous routing proposals [22, 30, 11] extend Chaum-Mixes [3] by forwarding traffic through a sequence of relays. Each relay is a single network endpoint. They attempt to ensure that the identity of the message source is never revealed to the destination, and the source and destination identities are hidden from relays and third-party observers. They achieve this by wrapping the payload and the sequence of relays through which a message is to be forwarded in layers of public key encryption, with one layer for each relay to be used. This requires that a set of relays be statically chosen at the beginning of a communication session. In general, if A sends a message M to B, then A defines a forwarding path that is a sequence of L relays $R_1, R_2, \ldots, R_L$. Each relay has a public/private key pair, where the public key of relay R_i is K_i. The message M is then sent encrypted in the form of $R_1 < R_2, < R_3, \ldots < R_L, < B, M >_{K_L} >_{K_{L-1}} \ldots >_{K_2} >_{K_1}$.

Successful end-to-end message delivery requires that every relay R_i in the forwarding path successfully decrypts its designated layer and forwards the message to the next relay. If the next relay has failed or is unreachable, then the message cannot be forwarded any further. When this occurs the source must discover the failure and then select a new set of live relays and resend the payload. Detecting failures in the routing path is made difficult because relays cannot send error messages to the anonymous source. This means that while these systems work in static and reliable networks, their performance degrades on less reliable wide-area links. They are also unlikely to function well on peer-to-peer and adhoc networks, where both end-point and link failure are observed regularly.

We propose a failure resilient anonymous routing system called Cashmere. Cashmere achieves resilience by using a set of distributed endpoints as a single virtual relay rather than a single endpoint. We refer to these endpoints as *relay groups*, and the forwarding path used in Cashmere is a sequence of *relay groups*. All members of a relay group share a public/private key pair. Layered encryption is still used on the forwarding path, using the public key of the relay group. Every member of the relay group has the ability to independently decrypt the next layer in the forwarding path. A forwarding path is valid as long as each relay group used in the forwarding path has at least one single live reachable member. While Chaum-Mixes route to the destination as the last hop, the destination in Cashmere is a member of any one of the relay groups on the forwarding path. The source randomly orders the relay groups to hide the destination relay group. When a message arrives at a member of a relay group, the receiver both anycasts the message to the next relay group and broadcasts the decrypted contents to all other members of the relay group. This ensures that if the destination is a member of the current group, it will receive the message.

Design Goals There are different types of anonymity [23]. Cashmere is designed to provide both *source anonymity* and *unlinkability* of source and destination. *Unlinkability* means that even if the source and destination can each be identified as participating in some communication, they can not be identified as communicating with each other. *Source anonymity* means that the identity of the source is hidden to all other nodes including the receiver. An attacker may be able to associate a set of messages with the same session but cannot determine the source, destination or the message payload. Provided the source does not divulge its identity in the message payload or collude with attackers, Cashmere provides both source anonymity and unlinkability even if the destination is controlled by an attacker. Cashmere can easily be extended to provide *destination anonymity*, where the destinations identity is hidden to all other nodes including the source, using an additional level of indirection.

Attack model We assume the attacker controls a fraction f of the nodes in the Cashmere network and these compromised nodes collude, sharing all information such as private keys. We assume a Byzantine failure model where compromised nodes can behave arbitrarily. The attacker can observe all messages sent over the network, regardless of whether the source or destination is controlled by the attacker, and there is zero latency for messages sent between compromised nodes.

The rest of this paper is structured as follows. We give an overview of related work and their limitations in Section 2. Next, we present the design of Cashmere in Section 3. We then discuss details of our current Cashmere implementation in Section 4. In Section 5, we analyze the level of anonymity in Cashmere and evaluate its security and performance using both simulation and measurements from an actual implementation. Finally, we outline future work and conclude in Section 6.

2 Related Works and Limitations

The original anonymous system redirected traffic through a centralized proxy [1]. Chaum [3] improved on this by using mix networks to create anonymous email, and inspired a number of subsequent systems [11, 24, 10, 7], including the Onion Routing system [22, 31]. Onion Routing relies on traffic redirection between a static set of dedicated onion routers that maintain pair-wise symmetric keys. To send a message, the source selects a set of currently active routers through which a message is forwarded. These requirements limit the scalability of Onion Routing, especially in environments with node churn. Tor [9] proposes using a directory server to maintain router information but this approach is also limited in scalability. It has also been shown that if the first or last router is compromised in an Onion Routing network, the source or destination is revealed [30].

Tarzan [11] also uses layered encryption and multi-hop routing. The source chooses a set of relays to act as a path and iteratively establishes a tunnel through these relays with symmetric keys between them. Hence, the creation of a tunnel incurs both significant computation overhead and delay. The tunnels are static and any relay failure requires formation of a new tunnel.

Crowds [23] and more recently AP3 [16] make use of probabilistic random forwarding. Crowds is limited in scalability because of its centralized admission control server, and has been shown to provide lower anonymity than Chaum-Mixes based systems [8].

Wright et al. [32, 33] have shown that relying on static forwarding paths impacts the anonymity properties of anonymous routing layers. They proposed a degradation attack applicable to Crowds, Onion Routing and other anonymizing systems that exploits the requirement to reconstruct the paths when they break due to node or link failure. During a long communication session, the path between source and destination is reconstructed many times, and each instance of the path must include the sender. After a large number of resets, the sender has much higher probability of being a path member than other nodes. Assume that the “first” attacker on each path (of the same session) logs its predecessor. After a number of path resets, the identity of the sender can be guessed with increasing probability.

Cashmere addresses these limitations by removing the reliance on static paths. By using flexible relay groups to maintain resilient long-lived paths, we improve performance by reducing path reconstruction time, and also reduce our vulnerability to the degradation attacks [32, 33] mentioned above. We gain these benefits with minimal loss to the level of anonymity attained compared to other Chaum-Mixes approaches.

3 Cashmere Architecture

Cashmere uses layered-encryption and multi-hop routing through relays. Instead of using a single node as a relay, Cashmere uses a set of nodes that act as a virtual relay, called a *relay group*. All members of a relay group share a common public/private key pair.

A forwarding path consists of a sequence of relay groups. *Any* member of a relay group is able to decrypt the forwarding path information for a message and forward the message to the next relay group. The membership of the relay group can change dynamically. As long as the relay group has at least one member, it is able to successfully relay messages. This makes Cashmere extremely resilient to node churn. A relay group is an anycast group, and the forwarding of a message in analogous to anycasting to the next relay group. Unlike in Chaum-Mixes where the destination is the last hop, in Cashmere the destination is a member of one of the relay groups.

Cashmere is built on a structured overlay, and we leverage this to both dynamically create and maintain the relays groups as well as for routing between relay groups.

3.1 Structured Overlay Networks

Structured overlay networks provide a scalable routing substrate for building resilient, large-scale decentralized systems [21, 26, 29, 34]. A structured overlay is composed of a set of nodes, where each *node* represents an instance of a participant in the overlay. The structured overlay maintains a large *k-bit identifier space*, *e.g.* k=160. Nodes are assigned *nodeIDs* uniformly at random from this space, generated and signed by an off-line central authority (CA).

Most structured overlays support *Key-Based Routing* (KBR) [6], enabling applications to route a message to any specified key selected from the identifier space. These overlays dynamically map each key to a unique live node in the overlay, the *root* node for the key. The root could be the node with the nodeID numerically closest or with the longest prefix match to the key.

Each node in a structured overlay maintains a *routing table* that typically contains $O(\log N)$ nodeIDs and IP addresses of other nodes in the overlay, where N is the number of nodes in the overlay. By using nodeID constraints when choosing nodes for their routing table, they can route messages in $O(\log N)$ hops.

Cashmere is designed to use a prefix-routing based structured overlay, like Tapestry or Pastry. Routing in such overlays requires that at each hop the message is forwarded to a node whose nodeID shares a longer prefix match than the current node's nodeID. Figure 1 shows an example of prefix routing. At each hop the prefix match between the current nodeID and the key increases by one digit. These protocols are resilient to node churn [4], and can route around a large number of link failures [35].

Cashmere is being used as an anonymous routing infrastructure. The attacker could attempt to compromise the structured overlay, and thus compromise anonymity layer built on top it. To address this, we assume the structured overlay is secured against malicious nodes using the techniques described in [2] and [14]. In this paper we do not address the issue of denial of service (DoS) attacks. In Cashmere, DoS attacks affect performance but not the level of anonymity. Finally, our design can tolerate a large proportion of malicious nodes, and anonymity can be increased by creating longer relay paths even if a large proportion of the overlay has been corrupted. We also generate sufficient cover traffic [1] in the network to prevent simple traffic analysis attacks.

3.2 Relay groups

Relay group membership management in Cashmere exploits properties of the identifier space maintained by prefix-routing based structured overlays. In particular, for each k-bit nodeID there are k unique prefixes. For example, the 6-bit nodeID `101011` has prefixes: `1`, `10`, `101`, `1010`, `10101` and `101011`. In general, if there are N nodes it is expected that $N/2^m$ will share the same m-bit prefix.

In Cashmere, each relay group has a *groupID* which is an m-bit identifier, where $1 \leq m \leq k$. A node is a member of that relay group if the groupID is a prefix of its nodeID. Since nodeIDs are randomly assigned, nodes in a relay group are a random subset of the overlay nodes and exhibit independent failure patterns. Each prefix requires a public/private key pair and all nodes that share that prefix need both the public and private key. We assume these are generated and distributed using an off-line CA. In general, a user wishing to contribute a node to the system must obtain from the CA a signed k-bit nodeID and the set of k public/private keys associated with its nodeID and must have access to all the public keys of the other prefixes. Each nodeID must be unique, so the public/private key for the k-bit prefix will be unique to this nodeID.

The structured overlay routes messages between relay groups. The groupID is used as the key as a message is routed using KBR. As the message is routed, the first node that receives the message and shares the groupID prefix processes the message on behalf of the relay group. This node is referred to as the *relay group root*. Therefore, routing a message to a groupID is effectively performing an anycast to the relay group members.

Generally, if node A wants to route a message to node B anonymously, it selects a random sequence of m-bit groupIDs that defines the set of relay groups and includes the m-bit prefix of B. These are used to construct a forwarding path, *i.e.* a sequence of relay groups the message routes through. Since A selects the groupIDs randomly, the path cannot be predicted by others. The value of m

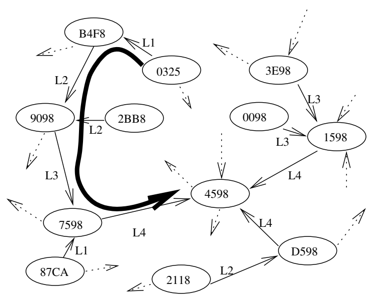

Figure 1: *Routing example in a structured overlay using prefix routing.* Node `5230` routes a message to the key `8954`. At each hop the message is forwarded to a node that shares a longer prefix with the key.

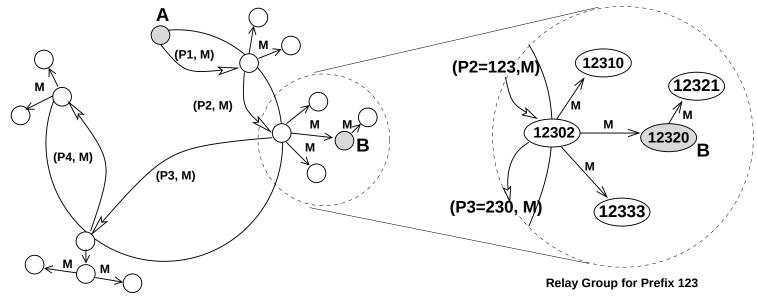

Figure 2: *A forwarding path from A to B composed of multiple relay groups.* Here a relay group is defined by a 3-digit prefix. At each relay group, the first node to receive the message broadcasts the message to all members of the group using *directed broadcast*. In the inset, node `12302` forwards the message to the rest of the relay group for prefix `123`.

controls the expected size of the relay group, and consequently the resilience against failures and malicious nodes. A encrypts the forwarding path in multiple layers using the public keys associated with each relay group. The overlay routes the message to the first relay group using its groupID. When any node matching the current prefix receives the message, it becomes the relay group root for that message, and uses the relay group's private key to decrypt the next layer of the path. This reveals the next groupID and the message is routed through the overlay to that prefix.

3.3 Decouple forwarding path and payload

Unlike other Chaum-Mixes based systems, Cashmere decouples the payload from the encrypted forwarding path, and encrypts the payload separately. This has the advantage that a source can reuse a forwarding path, avoiding multiple public key encryptions. The source caches the forwarding path, and only needs to perform a single public key encryption on each message using the destination's unique public key.

The source needs to encrypt each message payload such that it can only be decrypted by the true destination and such that each relay sees a different value for the payload (as do eavesdroppers). Suppose there are L relay groups in the forwarding path: $P_1, \cdots, P_L$ and the destination node B is in relay group P_d where $1 \leq d \leq L$. In order to encrypt the payload the source generates a symmetric key (R_i) for each relay group P_i, where $1 \leq i \leq L$. The source generates the payload:

$$\mathsf{Payload}_i = \begin{cases} \left\langle \mathsf{Payload}_{i+1} \right\rangle_{R_i} & 1 \leq i < d \\ \langle M \rangle_{\mathsf{PubKey}_B} & i = d \end{cases}$$

where $\langle M \rangle_{\mathsf{PubKey}_B}$ is the *real payload* encrypted by the destination's public key and $\langle \cdot \rangle_*$ indicates the content is encrypted using the key on the subscript. The source generates a forwarding path by:

$$\mathsf{Path}_i = \begin{cases} \langle \mathsf{Path}_{i+1}, P_{i+1}, R_i \rangle_{\mathsf{PubKey}_{P_i}} & 1 \leq i \leq L \\ \perp \text{ (termination)} & i = L+1 \end{cases}$$

The source then anycasts the tuple $[\mathsf{Path}_1, \mathsf{Payload}_1]$ to the first relay group P_1. In general, the i-th relay group root receives messages $[\mathsf{Path}_i, \mathsf{Payload}_i]$ from the previous relay group. The i-th relay group root uses the groups public key to decrypt the outer layer of Path_i, revealing Path_{i+1}, the identity of the next relay group, P_{i+1} and the symmetric key R_i. The i-th relay group root decrypts $\mathsf{Payload}_i$ using R_i, generating $\mathsf{Payload}_{i+1}$. Provided Path_{i+1} is not $\perp$ then the relay group root anycasts the tuple $[\mathsf{Path}_{i+1}, \mathsf{Payload}_{i+1}]$ to the next relay group P_{i+1}. During a single session, the source caches Path_1 and generates $\mathsf{Payload}_1$ for each message.

This process ensures that $\mathsf{Path}_i \neq \mathsf{Path}_j$ and $\mathsf{Payload}_i \neq \mathsf{Payload}_j$ if $i \neq j$. In particular, the source only encrypts the payload with the symmetric keys for the relay groups $R_1, \ldots, R_{d-1}$. The path has embedded within it the symmetric keys $R_1, \ldots, R_L$. At each of the relay groups $P_d, \ldots, P_L$ the payload will be decrypted using appropriate symmetric key, resulting in the forwarded payload being a random number. This ensures that $\mathsf{Payload}_i \neq \mathsf{Payload}_j$ if $i \neq j$.

However, there is no guarantee that when the message reaches P_d that the relay group root will be node B, as any member of a relay group can receive a message for its relay group. To ensure B receives the message, we multicast the payload to the entire relay group. If node X receives the message (thus becoming the relay group root for the message), then X decrypts the relay group's layer from the path in the message and decrypts the payload with the revealed R. X caches the map $\mathsf{Path}_i \leftrightarrow \langle \mathsf{Path}_{i+1}, P_{i+1}, R_i \rangle$ to reduce the computational load when further messages from the same ses-

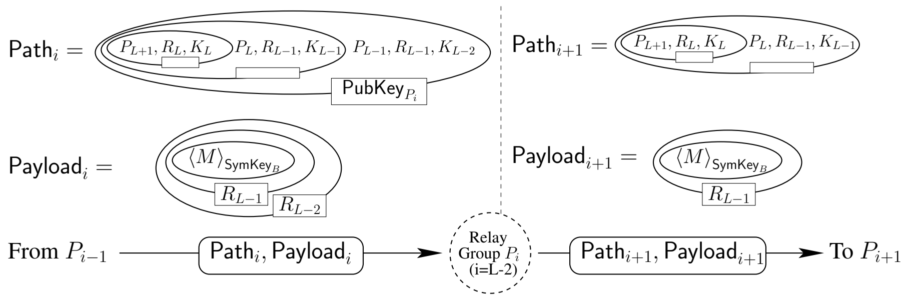

Figure 3: A detailed look at the path and payload components of a message, before and after processing at a relay group. The relay group root for P_i decrypts the layer around the path component to get P_{i+1}, R_i, K_i. It performs a symmetric decryption on the payload using R_i, and forwards the result to the relay group P_{i+1}.

sion are received. X forwards the message to the next relay group and broadcasts $\mathsf{Payload}_i$ to all members of the relay group (we discuss the exact mechanism in Section 4). No matter what position B's relay group is in the path, B will receive the message either directly or via a broadcast when the message routes to a member of its relay group. Only B will be able decrypt the payload successfully. An example of this Cashmere routing is shown in Figure 2.

The use of a broadcast has two implications addressed below; *(i)* that each node in a relay group has to perform an asymmetric decryption for each packet in a session; and *(ii)* malicious nodes can either drop messages or not broadcast them to the relay group. While such actions do not compromise anonymity they do negatively impact performance. We rely on end-to-end acknowledgments to detect failures and malicious nodes: if the source receives no acknowledgments, it can use timeouts to guide retransmission.

We eliminate the need to perform asymmetric encrypt/decrypt operations on the data payload by encrypting it using a symmetric key SymKey_B chosen when a source creates a path. In addition to the next relay group prefix, P_i, and a group session key, R_i, we embed another value K_i into the layered encrypted path. If destination B is in relay group d, then

$$K_d = \langle \mathsf{SymKey}_B | \mathsf{FLAG} \rangle_{\mathsf{PubKey}_B},$$

where $|$ means concatenation. All other K_i values are random numbers. Now the format of Path_i is changed to

$$\mathsf{Path}_i = \langle \mathsf{Path}_{i+1}, P_{i+1}, R_i, K_i \rangle_{\mathsf{PubKey}_{P_i}},$$

and M is no longer encrypted with PubKey_B but is now encrypted with SymKey_B, $\langle M \rangle_{\mathsf{SymKey}_B}$. Figure 3 illustrates the full mechanism.

Now relay group roots broadcast $\langle K_i, \mathsf{Payload}_i \rangle$ to all members in the relay group. B decrypts K_d and identifies FLAG, thereby knowing that it is the destination. Using SymKey_B B can decrypt M. All other members of this relay group cache K_i. For future packets in the same session, they remember they are not the destination node and without further decryption operations. B caches SymKey_B and associates it with K_i and therefore only needs to perform symmetric decryption for subsequent session payloads.

This also has the advantage that relay group roots can cache $\langle \mathsf{Path}_{i+1}, P_{i+1}, R_i, K_i \rangle$ if they have already forwarded messages for the same session. Relay group roots can identify messages using Path_i as the session ID, hence no asymmetric decryption is necessary.

3.4 Anonymous Reply Addresses

A destination can reply to a source without sacrificing source anonymity or requiring state to be stored in the relay groups in the forwarding path. The destination can reply to the source either a pre-formatted reply message (*e.g.* an acknowledgment) or a message containing an arbitrary payload. The reply message shares all of the performance and security benefits with the anonymous messages from source to destination.

Node A wishes to send an anonymous message to B and receive a reply. A creates a forwarding path to B as described, but also generates a return forwarding path from B to A. A does this by randomly selecting L relay groups ($P'_1, \ldots, P'_L$). The set of relay groups used in the return forwarding path may or may not intersect with the set of relay groups used in the forwarding path from A to B. A ensures that a relay group containing itself, $P_{d'}$, is included in the return path. A sends $\mathsf{ReplyAddrInfo}$ as part of the payload to B, where:

$$\mathsf{ReplyAddrInfo} = \left\langle \mathsf{Path}'_1, P'_1, \mathsf{SymKey}_A \right\rangle$$

$$\mathsf{Path}'_i = \begin{cases} \left\langle \mathsf{Path}'_{i+1}, P'_{i+1}, R'_i, K'_i \right\rangle_{\mathsf{PubKey}_{P'_i}} & 1 \le i \le L \\ \perp \text{ (termination)} & i = L+1 \end{cases}$$

$$K'_i = \begin{cases} k'_i & i \ne d' \\ \langle \mathsf{SymKey}_A | \mathsf{FLAG} \rangle_{\mathsf{PubKey}_A} & i = d' \end{cases}$$

where k'_i and R'_i are selected uniformly at random, $P'_{L+1} = \perp$. If B wants to send a payload M' to A, it sends Msg' as $[\mathsf{Path}'_1, \langle M' \rangle_{\mathsf{SymKey}_A}]$ to P'_1. While Msg' is created by B, it knows nothing about the path and the source. The root of each relay group P'_i decrypts Path'_i the same as in the forwarding path, while it encrypts $\mathsf{Payload}'_i$ with R'_i to get $\mathsf{Payload}'_{i+1} = \left\langle \mathsf{Payload}'_i \right\rangle_{R'_i}$. Node A who is located in relay group $P'_{d'}$ will receive message $\left\langle K'_{d'}, \mathsf{Payload}'_{d'} \right\rangle$, where $\mathsf{Payload}'_{d'}$ is the layered encryption of $\langle M \rangle_{\mathsf{SymKey}_A}$ by $R'_1, \ldots, R'_{d'-1}$. After A decrypts $K'_{d'}$ using PubKey_A, A can use SymKey_A to identify which session the reply belongs to, and thus the keys R'_i $(1 \leq i < d)$ to decrypt $\mathsf{Payload}'_{d'}$. All caching schemes used in the forwarding path also apply to the return path.

3.5 Selection of GroupID and Path Length

The final issue is how a source selects groupIDs for relay groups. Observation 1 shows the relation between the length of groupIDs and relay group sizes.

OBSERVATION 1: **(Distribution of Relay Group Sizes)** *Let N be the number of nodes in the overlay and nodeIDs are assigned to all nodes uniformly at random. The size of relay groups defined by a m-digit groupID is Poisson distributed with parameter $\rho = \frac{N}{2^m}$. The expected size of the relay group is ρ. [Proof omitted]*

A valid groupID requires that there exists at least one node that has the groupID as a prefix. As N is much smaller than the size of the nodeID identifier space, there will be many invalid groupIDs. From Observation 1, the probability that a groupID is valid is $p_1 = 1 - e^{-\rho}$. When a node forms a path by selecting groupIDs uniformly at random, the chance that the path contains only valid groupIDs is $(p_1)^L = (1 - e^{-\rho})^L$, where L is the number of relay groups used in the path. The expected number of tries to generate a valid path, one that is composed on only valid groupIDs, is $\frac{1}{(1-e^{-\rho})^L}$. Table 1 shows the average number of tries to generate a valid path is slightly larger than 1 under typical L and ρ values.

In Cashmere, nodes independently (without external communication) select per-session values of m (which determines ρ) and L to control tradeoffs between churn resilience, anonymity and overhead. We discuss this in Section 5.1. In general, choosing a value of between 3 and 5 for ρ, and a value of L between 4 and 8 provides a good combination of efficiency and resiliency. Because nodeIDs are uniformly distributed, nodes can locally estimate N using their routing tables. From Observation 1, a node can always get the average relay group size (ρ) it wants by selecting a proper prefix length m. The design of Cashmere removes the high cost of maintaining complete or near-complete overlay membership information.

	$L = 4$	$L = 5$	$L = 6$	$L = 7$	$L = 8$
$\rho = 4$	1.0767	1.0968	1.1173	1.1381	1.1594
$\rho = 5$	1.0274	1.0344	1.0414	1.0485	1.0556

Table 1: Average number of tries to get a valid path.

4 Implementation

We implemented Cashmere on top of FreePastry [12], a Java implementation of Pastry [26]. The implementation uses RSA (with 512-bit key length) and Blowfish (with 128-bit key length) as the asymmetric key and symmetric key ciphers, and uses the Cryptix [5] crypto library.

Applications use a simple Cashmere API. The source creates an *AnonymousChannel* object specifying a destination nodeID, and uses it to forward payloads. An application instance running on the destination node receives an up-call with the payload.

The Cashmere implementation ensures that relay group roots cache Path_i information and all nodes cache K_i as described in the previous section.

Our implementation performs relay group broadcast of $\langle K_i, \mathsf{Payload}_i \rangle$ using the leaf sets that are maintained by each node [26]. The leaf set is a set of pointers to the immediate l neighbors in the identifier space, where typical $l = 8$. If the leaf set does not contain all members of the relay group, nodes on the edge of the leaf set forward the message to their leaf set members. This recursive process continues until all members of the relay group have received the message.

One practical issue in the encoding of the path is that it is desirable for it to have the same length all along the forwarding path. This way no information about the route can be obtained by simply observing the size changes of the path onion. Previous work discussed these length-preserving Chaum-Mixes. A simple scheme is implemented in Mixmaster [18], and [17] presents a more sophisticated, provably secure scheme. Our prototype currently uses the basic layered encryption, and thus the path size decreases after each relay group. Changing the encoding scheme to preserve message length is straightforward and orthogonal to the design and performance of the overall system.

5 Analysis and Evaluation

5.1 Anonymity Measurement

We analyze two types of anonymity provided by Cashmere: source anonymity and unlinkability of source and destination. We quantify Cashmere's anonymity parameterized by:

- N: network size;

- f: fraction of malicious nodes in the network;
- ρ: average relay group size ($\rho = N/2^m$);
- L: number of relay groups in a path, the path length.

The parameter f has two implications: *(i)* the probability that compromised nodes are on the relay path; and *(ii)* the fraction of relay group private keys known by the attacker. For each compromised nodeID the attacker will know the relay group private keys for all prefixes associated with the nodeID. The probability that the attacker knows a m-bit prefix private key is $p_2 = 1 - e^{f\rho}$. The attacker can obtain prefix private keys either by compromising other nodes or through obtaining nodeIDs from the CA. We assume prefix private keys are leaking slowly, and the offline CA can slowly issue new prefix keys and revoke prior prefix keys over time. If the attacker knows the private key for a relay group we refer to the relay group as being compromised.

Our anonymity measurement follows the anonymity definition by Pfitzmann et al. [20]: "Anonymity is the state of being not identifiable within a set of subjects, the *anonymity set*." In a network with a finite set Ω ($|\Omega| = N$) of nodes, *ideal anonymity* is achieved when all nodes look equally likely to be the source or destination to an attacker, *e.g.* the anonymity set is Ω. In reality, based on information leaked from the system, some nodes look more likely to be the source or destination than others. That is, the attacker knows that the source (or destination) is in Ω_i with probability $\Pr(\Omega_i)$ [2], where $\Omega = \bigcup_i \Omega_i$. For example, the *worst anonymity* is the attacker identifies the source or destination as u_0; $\{u_0\}$ is assigned with probability 1 and $\Omega \backslash \{u_0\}$ with probability 0. We use the metric proposed in [8, 28] to measure the anonymity of our design as a proportion of the ideal entropy achievable in a given network. We briefly describe the entropy-based metric as follows:

DEFINITION 5.1. **(Entropy of a System).** *Ω is the (finite) set of all nodes in the network. Using knowledge of leaked information from the system, an attacker assigns each node u ($u \in \Omega$) a probability p_u as being the source or destination of a message. System entropy is defined as:*

$$H(\Omega) = -\sum_{u\in\Omega} p_u \log_2(p_u).$$

If we have ideal anonymity, all nodes look equal to attackers: $\forall u \in \Omega$, $p_u = \frac{1}{|\Omega|}$. The entropy of ideal anonymity is $H_m(\Omega) = \log_2(|\Omega|)$, which is the maximum entropy achieved in a network of $|\Omega|$ nodes.

DEFINITION 5.2. **(Anonymity of a System).** *The anonymity of a system is measured as:*

$$\frac{H(\Omega)}{H_m(\Omega)} = \frac{-\sum_{u\in\Omega} p_u \log_2(p_u)}{\log_2(|\Omega|)}.$$

Definition 5.2 shows that the anonymity of a system is measured by the real entropy of a system over the maximum (*i.e.* ideal anonymity) entropy the system could achieve: $0 \leq \frac{H(\Omega)}{H_m(\Omega)} \leq 1$.

The entropy definition above is more precise than the straightforward probability definition of *the probability that the attacker knows the sender or receiver*. For example, let us consider source anonymity in network of 10000 nodes. In an anonymity system AS_1 with attacker T:

- T discovers the source of 5% of messages;
- T can limit sources of 40% of messages to a small subset of nodes, e.g. 100;
- For the other 55% of messages, all nodes look equally likely to be a source to T.

In another anonymity system AS_2,

- T discovers the source of 5% of messages;
- For the other 95% of messages, all nodes look equally likely to be a source to T.

Using the probability that T knows the sender or receiver, both AS_1 and AS_2 have anonymity of 0.95. Using the entropy definition, the anonymity of AS_1 is:

$$0.05*0+0.40*\frac{-100*\log_2(\frac{1}{100})}{-10000*\log_2(\frac{1}{10000})}+0.55*1 = 0.552;$$

and anonymity of AS_2 is: $0.05 * 0 + 0.95 * 1 = 0.95$.

The entropy definition is more precise, capturing that AS_2 provides better anonymity. In AS_1 the attacker knows more information about the sources than in AS_2.

The anonymity of Cashmere is determined by ρ and L given the fraction of compromised nodes. Anonymity increases with larger values of L. Intuitively, the destination is hidden among all relay group members and ρ and L determine the number of nodes in all relay groups. However, as ρ increases, which means a shorter prefix is selected for groupID and the attacker has more chance to know consecutive relay groups, the anonymity decreases. Larger ρ also means more resilience and a higher relay group broadcast overhead. From analysis and experimentation, good typical values for ρ are between 3 to 5. In this section, we perform simulations with a network of $16,384$ nodes. GroupIDs have a prefix length of 12 bits, such that the expected size of relay groups $\rho = 4$ nodes. We compute unlinkability and source anonymity using the entropy definition. We first assume that attackers only see their own traffic, and simulate unlinkability and source anonymity given different parameters of (f, L). We then analyze the security of Cashmere against traffic analysis attacks.

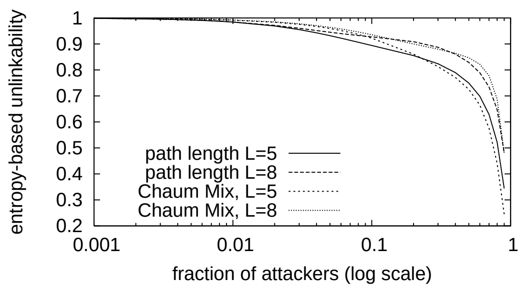

Figure 4: Anonymity measurement of unlinkability.

5.1.1 Unlinkability

In our simulations, the attacker gathers information observed from compromised nodes and maintains, for each pair of nodes (u_i, u_j), a probability p_{ij} that the pair are a source and destination.

Using the entropy definition, we can measure unlinkability using the relative entropy to ideal unlinkability:

$$\frac{\sum p_{ij}\log_2(p_{ij})}{N^2\cdot\left(\frac{1}{N^2}\log_2(\frac{1}{N^2})\right)}=\frac{-\sum p_{ij}\log_2(p_{ij})}{2\log_2 N}.$$

If the attacker believes u_i is the source with probability p_i and u_j is the destination with probability p_j, then $p_{ij} = p_i p_j$.

We assume the attacker determines the exact number of relay groups L used for a message [3]. We also assume the attacker knows a chain of n consecutive relay groups on the path of a message, each containing ρ_i nodes. Assuming there is enough cover traffic, the attacker cannot attribute discrete chains in the path to the same session, because the path onion and the observed payload are completely different at each relay group. Therefore, the attacker's knowledge about a message only comes from one consecutive chain on the relay path.

The source is indistinguishable from the *relay group root* of the immediately preceding relay group. When the first relay group root on the chain is non-malicious and known by the attacker, the attacker infers that the source is the first root with probability $\frac{1}{L-n+1}$ and the source is among all other non-malicious nodes with probability $1-\frac{1}{L-n+1}$. That is, for each non-malicious node u, the attacker assigns probability of u being the source as:

$$p_u = \begin{cases} \frac{1}{L-n+1} & \text{the first relay group root on the chain} \\ \left(1-\frac{1}{L-n+1}\right)\cdot\frac{1}{(1-f)N-1} & \text{otherwise} \end{cases}$$

When the first root on the chain is not known by the attacker or is malicious itself, all non-malicious nodes look equally to be the source, each with probability $\frac{1}{(1-f)N}$.

Let S be the set of nodes that are in the chain of relay groups known by the attacker, $|S| = \sum_{i=1}^{n}\rho_i$. The set of S is composed of both a set S_1 of malicious nodes and a set S_2 of non-malicious nodes: $S = S_1 \cup S_2$, $|S_1| = f|S|$ and $|S_2| = (1-f)|S|$. The expected number of nodes in all relay groups is $L\rho$. If the destination is among S_1, the attacker knows the destination and unlinkability becomes the source anonymity problem that we discuss in Section 5.1.2. If the destination is among S_2, the attacker infers that each node in S_2 is the destination with probability $\frac{1}{L\rho-f|S|}$ and the destination is among other nodes outside S with probability $1-\frac{(1-f)|S|}{L\rho-f|S|}$. That is, for each node u not in S_1, the attacker assigns the probability of u being the destination as:

$$p_u = \begin{cases} \frac{1}{L\rho-f|S|} & u\in S_2 \\ \left(1-\frac{(1-f)|S|}{L\rho-f|S|}\right)\cdot\frac{1}{N-|S|} & u\notin S \end{cases}$$

The number of relay groups compromised (*i.e.* n) is closely related to the fraction f of compromised nodes. If the compromised node was not the relay group root then the attacker would only learn the value of K_i and the payload, which is broadcast to the relay group. When the compromised node is the relay group root for a message the attacker also discovers the identity of the next relay group. If the compromised node is on the intermediate overlay hops between two relay group the attacker knows the previous or/and the next relay group root.

In Figure 4, we compare through simulation Cashmere's unlinkability metric to that of Chaum-Mixes approaches under different parameters of (L, f), ignoring eavesdropping and traffic analysis (see Section 5.1.3). In the simulation, we setup a relay path of length L, assign each node on the path and in the relay groups as compromised consistent with parameter f, count the probability of different cases that the attacker knows n consecutive groups, and compute the entropy in all cases. Then the entropy of the system is the average over all cases [8, 28].

The results show that Cashmere has similar anonymity to Chaum-Mixes. Cashmere even behaves better than Chaum-Mixes for small L and f near 1, when the whole Chaum-Mixes path is controlled by attackers with high probability while Cashmere still benefits from the anonymity among relay group members. We also measured how the level of unlinkability varies with network size and, as expected, unlinkability is largely independent of network size. Increasing network size from 20K nodes to 2 million nodes results in less than 3% variation in unlinkability. Reducing the network size to 64 provides similar unlinkability under the same f as large networks as long as ρ and L are set the same. Thus, bootstrapping Cashmere requires a small initial network of trusted nodes and then other nodes can join the network while maintaining the fraction of malicious nodes in the network as f.

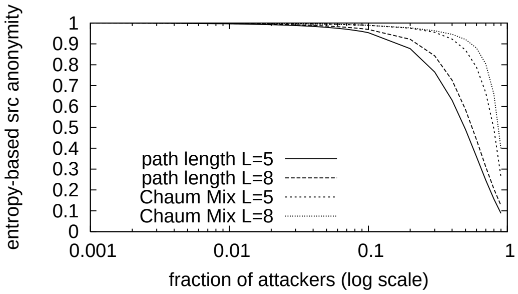

Figure 5: Source anonymity with anonymous messages.

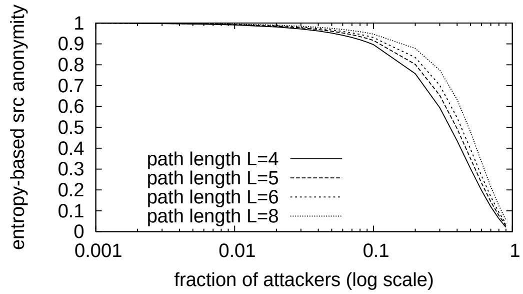

Figure 6: Source anonymity with anonymous channels.

5.1.2 Source Anonymity

In source anonymity, the destination colludes with other malicious nodes to find the source's identity. If Cashmere is being used for one-way communication (anonymous message) the attacker infers the first relay group root on the chain (which includes the destination's relay group) as the source with probability $\frac{1}{L-n+1}$ if the first root is non-malicious, where n is the length of the chain. Figure 5 compares the source anonymity of Cashmere with Chaum-Mixes, assuming no traffic analysis attacks, for one-way communication. We see that like Chaum-Mixes, Cashmere has high source anonymity when $f < 0.3$ and increasing L improves anonymity.

If Cashmere is being used for two-way communication (anonymous channel), the attacker has two ways to discover the source; *(i)* discover the first relay group root on the chain of consecutive relay groups which includes the destination's relay group (as for one-way), or *(ii)* the attacker compromises consecutive relay groups used on the return path from the destination to the source. Even if the attacker compromises all L of the return path relay groups, the attacker only knows the source is a member of one of these relay groups (the probability is the same as in Section 5.1.1).

Figure 6 shows the results for anonymous channels. The results show anonymous channels provide lower anonymity compared to anonymous messages due to the vulnerability of the return path. Finally, we also analyzed the impact of network size on source anonymity and, as before, increasing or decreasing the network size had negligible impact.

5.1.3 Robustness against Traffic Analysis

Our previous simulations disregarded the impact of traffic analysis. In practice, however, attackers may monitor part or all of the network traffic and use patterns to trace session paths. With each message, the same decoupled path component is sent from a relay root. For example, an attacker observes that a node u receives $[\mathsf{Path}_i, \mathsf{Payload}_i]$ and sends out $[\mathsf{Path}_{i+1}, \mathsf{Payload}_{i+1}]$ to u_2. Later it observes u receiving Path_i with a different payload, and sending Path_{i+1} with other another payload to u_2. The attacker can then recognize all messages with path component Path_{i+1} as parts of a session involving u and u_2. We simulate the robustness of Cashmere in unlinkability and source anonymity against an attacker observing increasing amounts of network traffic. There are two attacker models: *(i)* the attacker analyzes a fixed fraction of all network traffic, e.g. 0%, 90%, 100%, etc.; or *(ii)* the attacker analyzes a fraction, f_t, of traffic proportional to the fraction of malicious nodes (f) in the network. For example, 10% of malicious nodes can analyze 10% of all traffic. The second is a more realistic model.

We simulate unlinkability and source anonymity for anonymous channels (since it is weaker than anonymous messages), and plot the results in Figures 7 and 8, using parameters $L = 6$. We see that Cashmere is vulnerable to traffic analysis if the attacker observes a significant portion ($> 90\%$) of all network traffic. But Cashmere can still provide high levels of anonymity in the more realistic proportional traffic analysis model.

Cashmere can completely disable traffic analysis attacks with a small modification. Each node in the underlying structured overlay can exchange symmetric keys with peers in its routing table. This sets up secure channels between all node pairs and encrypts all messages using a symmetric cipher. Thus source anonymity and unlinkability are protected against the strongest attacker who can monitor all network traffic. The key-exchange cost is done once per lifetime of a node, in contrast to previous approaches that require per-session key exchanges [11]. Additionally, the small ($O(\log N)$) number of neighbors for each node limits number of key exchanges, whereas approaches like Onion Routing require $O(N^2)$ keys. Finally, the secure channel is established lazily when the first message is routed through that link.

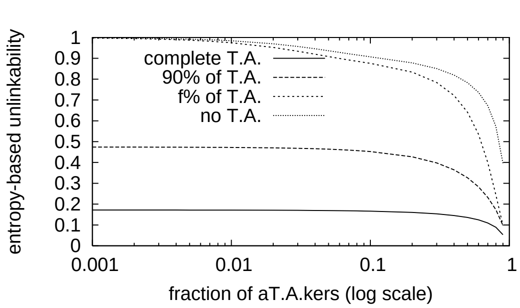

Figure 7: Unlinkability in anonymous channels under different types of traffic analysis (T.A.) with $L = 6$.

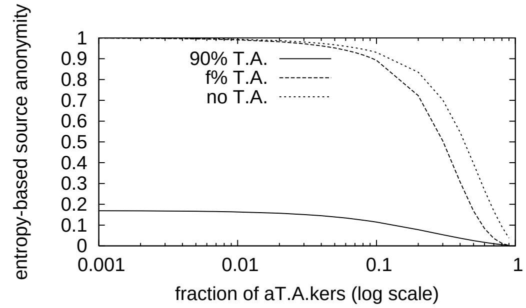

Figure 8: Source anonymity in anonymous channels under different types of traffic analysis (T.A.) with $L = 6$.

5.2 Resilience and Fault-tolerance

Previous anonymous systems use single nodes as relays. Nodes joining and failing in the system can lead to forwarding paths failing. Here we examine the resilience of Cashmere to node churn and intermittent link failures.

We refer to the time between a forwarding relay path is formed and its failure as the *relay path duration*. When a path fails, the sender needs to detect the failure via end-to-end timeouts and establish a new path. If relay path durations are too short, path construction time will dominate. Nodes will constantly be rebuilding failed paths and unable to deliver a message to a destination. Frequent path reconstruction also makes the layer more vulnerable to the degradation attack [32] discussed in Section 2.

In contrast, in Cashmere a relay is usable as long as at least one single node in the network has the relay group's groupID as a prefix. Changes in the membership of the relay group due to node joining and failing are transparent. We first compare the path duration and resilience of Cashmere to previous works.

5.2.1 Churn-resilience

Measurements on real systems have shown that peer-to-peer networks exhibit high node churn [27, 13]. Since most anonymous routing layers are implemented on overlay networks, they must be resilient to high node churn in order to be useful.

Previous studies [25, 13, 27] use *session time* as a metric of churn-rate. We approximate this using an exponential distribution with parameter μ. This churn model is consistent with those used in previous studies of the effect of churn in peer-to-peer systems [15, 25]. Our network model is as follows:

- The network is a finite set (Ω) of nodes, $N = |\Omega|$. The network size is stable, that is, node joins and failures are equal.
- *Session time* is exponentially distributed with parameter μ, meaning node failure is a Poisson process with rate $N\mu$. The mean session time is $\frac{1}{\mu}$ and the median session time is $\frac{\ln 2}{\mu}$.
- *Node arrivals* is a Poisson process with rate λ, where $\lambda = N\mu$.

Previous measurements [27] of file sharing systems suggest median session times of $\frac{\ln 2}{\mu} \approx 60$ min which we used for these experiments.

Figure 9 shows the expected path durations for forwarding paths using relay groups compared to using single nodes as relays. As expected, the use of relay groups increases the expected path duration exponentially, making Cashmere much more resilient to node churn.

5.2.2 Tolerance to Intermittent Failures

We now simulate Cashmere's tolerance to short-term intermittent failures. We model the mean time between failure (MTBF) as $\frac{1}{\lambda_1}$ and mean time to repair (MTTR) as $\frac{1}{\mu_1}$. We assume the failure is a Poisson process with failure event rate λ_1 and time to repair is exponential distributed with parameter μ_1. We assume MTBF $\frac{1}{\lambda_1} = 200$min, and MTTR $\frac{1}{\mu_1} = 5$min.

Figure 10 shows that Cashmere completely masks all intermittent network failures: the expected path duration is more than 10^6 minutes (about 40 days) when we set $\rho = 4$. This is an improvement of several orders of magnitude over previous node-based approaches.

5.2.3 Simulation on Kazaa Measurements

We examine how Cashmere's good path duration properties translate into stability for a real application. We simulate the fetching of objects in a file-sharing application, and examine the number of path repairs required during the object fetches. We model node churn and intermittent failures using parameters from the previous two sections. The distribution of object download times is long-tailed and generated using measurements from the Kazaa network. The Kazaa data [13] has distributions of down-

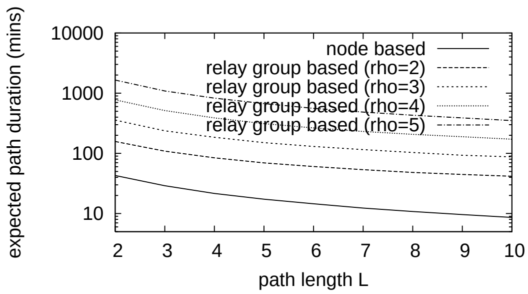

Figure 9: Comparing expected durations of node-based relays and "relay group"-based paths. (Note: ρ is shown as "rho", y-axis is log-scaled.)

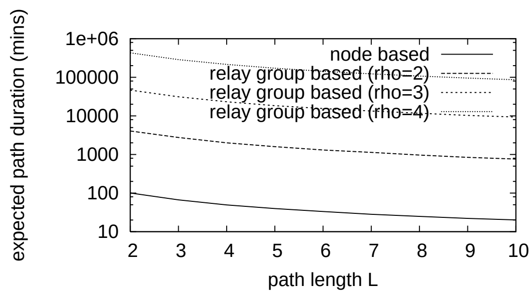

Figure 10: Comparing expected durations of node-based paths and "relay group"-based paths with intermittent failures. (Note: ρ is shown as "rho"; y-axis is log-scaled.)

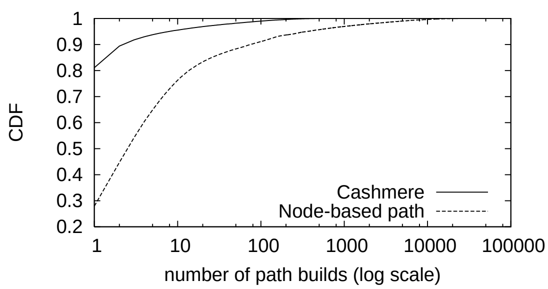

Figure 11: CDF distribution of number of path builds using all downloads in the Kazaa trace, comparing Cashmere (L=6, ρ=5) and node-based relays.

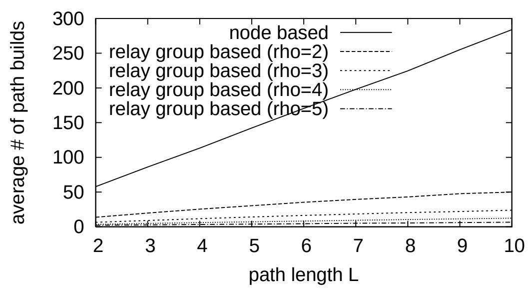

Figure 12: Average number of path builds for small object (10M) downloads using Kazaa data, comparing Cashmere and node-based relays.

load times for small objects (10MB) and large objects (100MB).

We simulate $100,000$ object download sessions on top of both node-based relays and Cashmere's group-based relays. Both systems use relay paths of length $L = 6$, and Cashmere uses average relay group size $\rho = 5$. Using object download times from the Kazaa data, Figure 11 shows the distribution of expected frequencies that each download needs to construct the relay path. It shows that 81% of these small object download sessions using Cashmere would not require any path rebuilds (*i.e.* number of path builds is 1) and no sessions require more than about 500 rebuilds. This compares to 28% using node-based relays, and 10% of all sessions requiring between 100 and 25000 path rebuilds. The maximum number of path builds is very large (*i.e.* 500 and 25000) because Kazaa object download times are long-tail distributed where some objects take extremely long time to download.

The average number of path builds under different parameters (L, ρ) for small object downloads are shown in Figure 12. Clearly, increasing relay group size increases path duration significantly, and Cashmere provides more than an order of magnitude improvement over node-based approaches. Measurements for large file downloads are nearly identical and omitted for brevity.

5.3 Cost and Performance Comparison

In this section we analyze the relative costs in operating Cashmere compared to previous node-based relay approaches. We observe that the operating costs of node-based relay path systems include:

1. Communication costs to maintain knowledge of candidate relays nodes.
2. Bandwidth cost in forwarding messages.
3. Computational costs to construct the relay path at the source and to decrypt messages at intermediate relay nodes.

We first examine communication costs in network maintenance and relay discovery. In node-based relay approaches, nodes are expected to actively maintain information about the other nodes in the network, with a total cost of $O(N^2)$. In contrast, Cashmere decouples maintenance and relay discovery, and relay discovery requires no communication. Nodes estimate the number of nodes in the network by examining their local routing tables, and choose an appropriate prefix length to establish relay groups of average size ρ. Nodes then choose

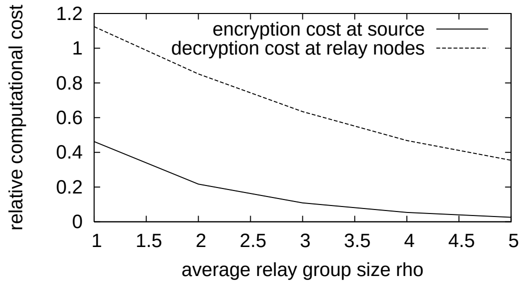

Figure 13: The relative aggregate computational cost over time compared to node-based approaches in a dynamic network. (Note: ρ is shown as "rho" in the figure.)

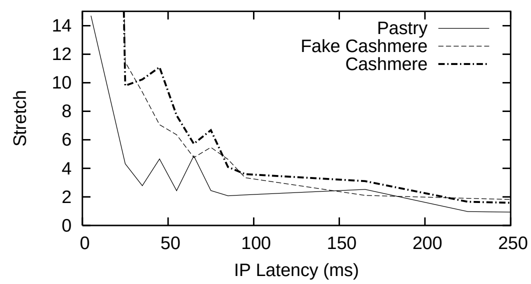

Figure 14: Stretches: Cashmere latency, fake Cashmere latency and Pastry latency vs. IP latency.

random prefixes of the desired length as relay groupIDs. Cashmere relies on the underlying structured overlay, and hence has a total cost of $O(N \log N)$.

However, Cashmere incurs a higher bandwidth cost to gain resilience. Total number of messages sent is $O(\rho L)$ while node-based approaches requires $O(L)$. The extra messages are required to perform the per-relay group broadcast of the payload, and do not adversely impact end-to-end latency or throughput at the overlay layer. This broadcast traffic does contribute to a node's cover traffic that it has to generate.

We now examine computational cost. High per-message computation is often seen as a key obstacle to the wide-spread deployment of Chaum-Mixes based systems. Given a path of length L, a Chaum-Mixes source node performs L asymmetric encryption operations on every message. In addition, each node on the path performs one asymmetric decryption per message that it forwards. The high cost of asymmetric cryptographic operations limits the message send rate at the source and the message forwarding rate at intermediate nodes.

Optimizations have been proposed to reduce computation for session-based communication on Chaum-Mixes by using symmetric key encryption for payload messages and amortizing asymmetric crypto operations across an entire session. Both Tarzan [11] and our solution fall into this category.

Assume the cost of asymmetric encryption and decryption are C_e and C_d respectively. For each relay group path, Cashmere incurs computational cost that includes encryption cost of $L \cdot C_e$ at the source, decryption cost of $2C_d$ at relay group root, decryption cost of C_d at each relay group member, and additional operations to refresh caches after relay group root failures. However, these cost are amortized over a much longer path duration than node-based systems and dwarfed by the cost of rebuilding paths in node-based systems.

Based on previous results of expected durations, Figure 13 plots the cost of our "relay group"-based approach relative to that of node-based session solutions on a realistic network. As ρ increases, the path duration increases and the per-session cost drops. For $\rho = 4$, the encryption cost at the source in Cashmere is roughly 5.37% of the cost at source nodes in node-based solutions. The aggregate decryption cost at relay group members in Cashmere is 46.83% of the cost at intermediate nodes in node-based solutions. The reduction in encryption computation is from amortizing the one time path setup costs across the long path durations of Cashmere. The reduction in decryption costs is from per-node caching of the path component and whether a node is the destination, and reducing the number of asymmetric crypto operations to just one per session for nodes who are not the destination.

5.4 Implementation Measurements

We ran experiments to determine the latency, throughput and computational overheads of Cashmere.

We deployed and evenly distributed 128 Cashmere nodes on 32 machines from PlanetLab that are geographically distributed all over the United States. We define groupIDs to be 5-bit prefixes, so relay groups have average size of 4 nodes. We measure latency in:

- *Cashmere*: End to end latency of Cashmere routing across 4 relay groups;
- *Fake Cashmere*: End to end latency of Cashmere routing across 4 relay groups, removing cryptographic computation;
- *Pastry*: The latency of routing via Pastry directly from source to destination;
- *IP*: Direct IP latency.

Message payloads are 24 bytes long. The latency is measured using round trip time (RTT), by sending messages from one node to all other nodes with each repeated 10 times.

We show the average latency in Cashmere, Fake Cashmere, Pastry vs. direct IP latency in Figure 14. The "stretch" is computed as each sample of Cashmere/Fake

Cashmere/Pastry latency divided by average IP latency for the same destination. To plot the graph, we put all stretch samples into bins of 10ms intervals of average IP latency. Figure 14 shows that the stretches decrease while the IP latency between source and destination increases. For a pair of end nodes that are very close to each other (i.e. < 50ms), Cashmere stretches are about two times of Pastry. The extra delays introduced by the Cashmere layer is significant compared to small IP latency values. Most samples of IP latency are from 50ms to 250ms. In this range, Cashmere stretches are between 1.9 to 5.5, which is quite close to Pastry (2.1 to 4.8). This means Cashmere layer introduces a relatively small delay on the overlay. Comparing stretches between Cashmere and fake Cashmere shows that delay caused by cryptographic computation in Cashmere is negligible. This is attributed to no per message asymmetric encryption/decryption in Cashmere. We also measured that the average number of IP messages per Cashmere message is 19.54 and the average number of IP messages per Pastry message is 1.54. The larger number of IP message comes from the relay and broadcast messages in Cashmere.

To measure computation cost, we utilize FreePastry's network emulation capabilities. We created 64 virtual FreePastry nodes inside the same Java virtual machine on a 2.4Ghz Pentium IV PC. The virtual nodes are connected together using local loopback (called "direct" network in FreePastry) network transport. There is no CPU contention between the nodes because the emulation is event-driven and at most one virtual node is running at a time. Cashmere is set up similarly as above. We obtain highly accurate time measurements by calling the `RDTSC` instruction supported by the Pentium architecture via Java Native Interface (JNI).

In the first experiment, we approximate throughput of relay group roots by measuring per-message latency across 1000 random source-destination pairs. For each source and destination pair, we send a single message to set up the path and allow relay group roots to set up their caches, then measure the latency taken to process a second payload message. We then approximate the throughput as $\frac{1}{latency}$. Table 2 shows the results for forwarding throughput of relay group roots for different message sizes.

In the second experiment we measure the computational overheads for the source, the relay group root nodes, the non-root relay group nodes and the destination, for both the first and subsequent messages. 1000 empty messages are sent from random source to destination with and without the routes already set up. Table 3 summarizes the results, showing the average CPU time incurred per node role with the standard deviation in brackets. The first message invokes RSA on each hop and therefore is relatively expensive. The subsequent messages to the same destination, using the same forwarding path, utilize cached routing information on each node. Therefore they only invoke Blowfish which is less expensive.

Msg Size (B)	Msg/second	Throughput (Mb/s)
128	1370	1.337
1024	1160	9.063
4096	855	26.72
16384	386	48.25

Table 2: Message forwarding rate and effective throughput for different message sizes of relay group root nodes.

	First Msg	Subsequent Msg
Source	8.21 (5.3)	0.73 (0.39)
Relay Group Root	27.5 (11.8)	0.22 (0.10)
Non-root Group Member	4.73 (347)	0.001 (0.05)
Destination	7.19 (1.87)	0.18 (0.03)

Table 3: CPU time (ms) spent by each class of node routing an empty message using Cashmere. Standard deviation shown in parentheses.

We also evaluated the space overhead during the experiment. At the source nodes the overhead for each message is 456 bytes for the path element and any necessary padding bytes to round the payload to RSA block sizes (64 bytes).

6 Conclusion

We present Cashmere, a resilient anonymous routing infrastructure that leverages the flexible anycast routing inherent in structured overlay networks to significantly improve path durations compared to node-based relay approaches. Cashmere also decouples the encrypted path component of each session from the payload, and uses symmetric session keys to encrypt message payloads. Anonymous source nodes in Cashmere can choose their own per-session parameters to tradeoff between anonymity, resilience and computation overhead.

We compare Cashmere to previous node-based Chaum-Mixes approaches through analysis and simulation. We find that Cashmere provides similar anonymity properties while providing one to two orders of magnitude improvement in path durations under both node churn and intermittent failures. This translates into significantly lower path reconstructions across an anonymous application session. Performance optimizations in Cashmere avoid asymmetric crypto operations, resulting in lower per-session computation costs compared to other session-based Chaum-Mixes approaches. Finally, we provide measurements of a real Cashmere deployment and show that it provides reasonable throughput while incurring a small latency overhead over structure overlay routing.

Ongoing work on Cashmere includes issues related to key management and key revocation in particular. We are also interested in better understanding the impact of network dynamics on key discovery. A straight-forward yet very useful extension to Cashmere is to support anonymous object location in DOLR [6, 34] overlays like Pastry and Tapestry. Finally, we are working on a stable wide-area deployment on PlanetLab and a software package for public release.

Acknowledgements

We would like to thank the anonymous reviewers for their comments and Vern Paxson our shepherd. We also thank Stefan Saroiu and Krishna Gummadi for providing us with data on filesharing session statistics.

Notes

[1]Cashmere only requires each node generates a small amount of traffic. When the real traffic is not sufficient, nodes send out dummy messages as cover traffic.

[2]Nodes in Ω_i are equal, each with probability $\frac{\Pr(\Omega_i)}{|\Omega_i|}$ to be the source (or destination).

[3]This is a worst case assumption. In reality the attacker can only estimate this by monitoring certain network latencies and system overheads. For example, the more relay groups are used, the more computation a source will perform.

References

[1] The anonymizer. Anonymizer.com.

[2] CASTRO, M., DRUSCHEL, P., GANESH, A., ROWSTRON, A., AND WALLACH, D. S. Security for structured peer-to-peer overlay networks. In *Proc. of OSDI* (Dec 2002).

[3] CHAUM, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM 24*, 2 (1981).

[4] COSTA, M., CASTRO, M., AND ROWSTRON, A. Performance and dependability of structured peer-to-peer overlays. In *Proc. of DSN* (Jun 2004).

[5] CRYPTIX TEAM. Cryptix. `http://www.cryptix.org/`.

[6] DABEK, F., ZHAO, B., DRUSCHEL, P., KUBIATOWICZ, J., AND STOICA, I. Towards a common API for structured P2P overlays. In *Proc. of IPTPS* (Feb 2003).

[7] DANEZIS, G. Mix-networks with restricted routes. In *Proc. of PET* (Mar 2003).

[8] DIAZ, C., SEYS, S., CLAESSENS, J., AND PRENEEL, B. Towards measuring anonymity. In *Proc. of PET* (Apr 2002).

[9] DINGLEDINE, R., MATHEWSON, N., AND SYVERSON, P. Tor: The second-generation onion router. In *Proc. of the USENIX Security Symposium* (Aug 2004).

[10] DINGLEDINE, R., AND SYVERSON, P. Reliable MIX cascade networks through reputation. In *Proc. of FC* (Mar 2002).

[11] FREEDMAN, M. J., AND MORRIS, R. Tarzan: A peer-to-peer anonymizing network layer. In *Proc. of CCS* (Nov 2002).

[12] Freepastry. `http://freepastry.rice.edu/`.

[13] GUMMADI, K. P., DUNN, R. J., SAROIU, S., GRIBBLE, S. D., LEVY, H. M., AND ZAHORJAN, J. Measurement, modeling, and analysis of a peer-to-peer file-sharing workload. In *Proc. of SOSP* (Oct 2003).

[14] HILDRUM, K., AND KUBIATOWICZ, J. Asymptotically efficient approaches to fault-tolerance in peer-to-peer networks. In *Proc. of DISC* (Oct 2003).

[15] LIBEN-NOWELL, D., BALAKRISHNAN, H., AND KARGER, D. Analysis of the evolution of peer-to-peer systems. In *Proc. of PODC* (Jul 2002).

[16] MISLOVE, A., OBEROI, G., POST, A., REIS, C., DRUSCHEL, P., AND WALLACH, D. S. Ap3: Cooperative, decentralized anonymous communication. In *Proc. of SIGOPS European Workshop* (Sep 2004).

[17] MÖLLER, B. Provably secure public-key encryption for length-preserving chaumian mixes. In *Proc. of CT-RSA* (Apr 2003).

[18] MÖLLER, U., COTTRELL, L., PALFRADER, P., AND SASSAMAN, L. Mixmaster Protocol — Version 2. Draft, Jul 2003.

[19] NGAN, T., WALLACH, D., AND DRUSCHEL, P. Enforcing fair sharing of peer-to-peer resources. In *Proc. of IPTPS* (Feb 2003).

[20] PFITZMANN, A., AND KÖHNTOPP, M. Anonymity, unobservability and pseudonymity. In *Proc. of the Intl. Workshop on the Design Issues in Anonymity and Observability* (Jul 2000).

[21] RATNASAMY, S., FRANCIS, P., HANDLEY, M., KARP, R., AND SCHENKER, S. A scalable content-addressable network. In *Proc. of SIGCOMM* (Aug 2001).

[22] REED, M. G., SYVERSON, P. F., AND GOLDSCHLAG, D. M. Anonymous connections and onion routing. *IEEE JSAC 16*, 4 (May 1998).

[23] REITER, M., AND RUBIN, A. Crowds: Anonymity for web transactions. *ACM Trans. on Inf. and Syst. Secur. 1*, 1 (Jun 1998).

[24] RENNHARD, M., AND PLATTNER, B. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proc. of WPES* (Nov 2002).

[25] RHEA, S., GEELS, D., ROSCOE, T., AND KUBIATOWICZ, J. Handling churn in a DHT. In *Proc. of USENIX* (Jun 2004).

[26] ROWSTRON, A., AND DRUSCHEL, P. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of Middleware* (Nov 2001).

[27] SAROIU, S., GUMMADI, P. K., AND GRIBBLE, S. A measurement study of peer-to-peer file sharing systems. In *Proc. of MMCN* (Jan 2002).

[28] SERJANTOV, A., AND DANEZIS, G. Towards an information theoretic metric for anonymity. In *Proc. of PET* (Apr 2002).

[29] STOICA, I., MORRIS, R., KARGER, D., KAASHOEK, M. F., AND BALAKRISHNAN, H. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proc. of SIGCOMM* (Aug 2001).

[30] SYVERSON, P., TSUDIK, G., REED, M., AND LANDWEHR, C. Towards an analysis of onion routing security. In *Proc. of PET* (Jul 2001).

[31] SYVERSON, P. F., GOLDSCHLAG, D. M., AND REED, M. G. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy* (May 1997).

[32] WRIGHT, M., ADLER, M., LEVINE, B. N., AND SHIELDS, C. An analysis of the degradation of anonymous protocols. In *Proc. of NDSS* (Feb 2002).

[33] WRIGHT, M. K., ADLER, M., LEVINE, B. N., AND SHIELDS, C. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur. 7*, 4 (2004).

[34] ZHAO, B. Y., HUANG, L., RHEA, S. C., STRIBLING, J., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Tapestry: A global-scale overlay for rapid service deployment. *IEEE J-SAC 22*, 1 (Jan 2004).

[35] ZHAO, B. Y., HUANG, L., STRIBLING, J., JOSEPH, A. D., AND KUBIATOWICZ, J. D. Exploiting routing redundancy via structured peer-to-peer overlays. In *Proc. of ICNP* (Nov 2003).