2005

Phyllo: A Peer-to-peer overlay security framework

William Heinbockel

Minseok Kwon

Recommended Citation

Phyllo: A Peer-to-Peer Overlay Security Framework, Proceedings of the First Workshop on Secure Network Protocols. November 2005.

Phyllo: A Peer-to-Peer Overlay Security Framework

William Heinbockel
The MITRE Corporation
Bedford, MA 01730
Email: heinbockel@mitre.org

Minseok Kwon
Department of Computer Science
Rochester Institute of Technology
Rochester, NY 14623
Email: jmk@cs.rit.edu

Abstract

Despite the success of peer-to-peer systems, a majority of their overlay architectures are vulnerable to exploitation. Some of the features for improved performance have created security holes that attackers can breach to gain control of the network. De facto network security solutions (e.g., trusted servers, encryption, and firewalls) offer little assistance, as they are often not compatible with the open and decentralized structure of peer-to-peer networks.

To address overlay security problems, we propose Phyllo, a node-partitioning framework that isolates untrusted nodes from the core network. Yet, the isolated nodes can still participate in peer-to-peer communications. Our partitioning scheme also allows nodes to move between partitions, while introducing marginal performance overheads. Our experimental results indicate that Phyllo supports more reliable message delivery in the presence of malicious nodes.

1. Introduction

Imagine the damages an attacker may cause if he controls an entire network. The attacker can filter and manipulate all inbound and outbound network traffic. Within peer-to-peer (P2P) systems, if there exist only a small percentage of attackers throughout the network, then the potential for malicious activity is relatively low. As the fraction of attackers increases, however, the attackers may potentially dominate larger portions of the network. The potential for the domination escalates even further if these attackers collude.

Peer-to-peer overlay networks (e.g., Chord [12] and Pastry [10]) currently offer few solutions for handling potential attacks [14, 2, 11]. Most overlays assume that malicious nodes will never comprise a considerable percentage of the network. To the best of our knowledge, however, there are few efficient and reliable solutions for security issues in overlay networks.

We introduce our framework, *Phyllo*, to reduce the detrimental effects that untrusted nodes might cause on overlay networks. The basic idea of Phyllo is to partition the fundamental overlay structure in order to isolate damages from adversaries within a single partition. Promotion and demotion protocols determine which nodes are transferred between partitions. A customizable evaluation platform is available for each network to specify the conditions for this transfer. Our framework can be easily implemented on popular overlay network systems such as Pastry. Phyllo requires at most two additional routing hops, and minimal processing and communication overheads.

The remainder of this paper is organized as follows. Section 2 gives motivation for our framework. In Section 3, we discuss the basic architecture of Phyllo including its routing mechanism and promotion/demotion protocols. In Section 4, we present results from the experiments using our implementation of Phyllo on Pastry. Section 5 gives an overview of related work. Finally, we conclude the paper and summarize future work in Section 6.

2. Motivation

The primary motivation behind our framework is the architectural security problems in overlay networks detailed in [14, 2, 11]. An attacker may exploit such problems to control certain routing paths, or cause unnecessary communication and routing overheads. Our architecture aims at the five main security issues outlined in [14] and [11]:

- **ID assignment problem** occurs when an attacker exploits the assignment of node identifiers (IDs) to obtain a specific node ID. Networks such as Pastry, which can use a node's IP address to cluster IDs based on their locality, allow attackers to control specific portions of the network.

- **Message forwarding** is an overlay-centric version of the Byzantine Generals Problem. Since peer-to-peer networks rely solely on other peers for routing, we should ensure that a message is properly routed and not be modified in transit.
- **Routing table maintenance** in dynamically routed overlays such as Pastry, is susceptible to malicious nodes propagating bad routing information. Without any extra detection mechanism, all incoming data are assumed to be reliable in peer-to-peer overlay networks.
- **Lookup attacks** involve a node directing lookup requests to wrong or invalid nodes, ultimately returning an incorrect response.
- **Denial of service attacks** can easily be performed against overlay networks. These attacks are, however, not considered as a principal motivation in our framework.

The taxonomy described in [8] offers an overview of the procedures that peer-to-peer networks should adopt to alleviate rational attacks as described above. With such attacks, users act in their own interest rather than the interest of the entire network. We recommend using incentives instead of punishing new users to protect the network. These suggestions are taken into account to design a deployable secure overlay architecture.

3. Phyllo Framework

The objective of Phyllo is to provide a flexible security interface for peer-to-peer overlay networks. In this section, we describe the details of the Phyllo framework. We first define a few terms and the basic architecture. Then, we discuss the routing algorithm and promotion/demotion protocols.

3.1. Architecture

Phyllo utilizes a ring-like structure similar to Pastry [10] and Chord [12]. The framework is, however, not limited to only these two networks. The information presented here can easily be adapted to any structured overlay network with unique node identifiers.

To minimize the effect of malicious nodes, we suggest partitioning the nodes into leaf-like structures around the circumference of the ring (Figure 1). The nodes that comprise the major partition (original ring) are referred to as *major nodes* (e.g., nodes `00`, `42`, and `91` from Fig. 1), while the remainder of the nodes, existing in the minor partitions, are *minor nodes* (nodes `1C`, `24`, `9C`, etc.). We also define

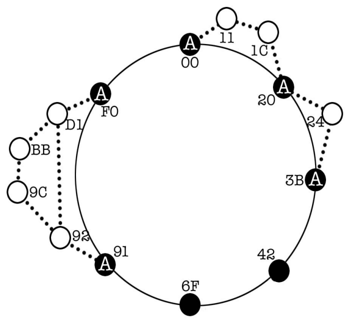

Figure 1. Phyllo topology with major (black nodes) and minor (white nodes) partitions. Anchor nodes are shown as the black nodes with an `A`. The nodes are assigned node IDs in an ID space of 2^8.

anchor nodes, which are the two major nodes with the IDs that bound the ID range of a minor partition, in every minor partition (nodes `00`, `20`, `3B`, `91`, and `F0`). Anchor nodes belong to both the major partition and the minor partitions connected to the anchor nodes. Anchor nodes help filter incoming messages based on their senders. This separation essentially prevents malicious nodes from forcibly escalating from minor partitions to the major partition.

The partitioned ring topology employs a single node ID space. Initially, a new node is positioned in a minor partition as to preserve the original ordering of the ID space. Both major and minor nodes are updated with only the routing information for their partitions. For example, if a node (e.g., node `1C`) broadcasts out new routing information, that broadcast is restricted to only its partition (i.e., node `11` and the anchors: `00` and `20`). An anchor node maintains additional routing tables for the adjoined minor partitions. A node may move from a minor to the major partition when the node becomes qualified. Details of the promotion and demotion protocols are discussed later in Section 3.3.

Phyllo distinguishes two message types. An *application message* is the message sent by a peer-to-peer application (e.g., search, response, and file transfer messages for a file-sharing application). The other message type is an *overlay message*, which supports overlay network management independent of applications (e.g., join and routing update messages). The distinction of the two message types helps constrain overlay messages within their originating partition. An anchor node ensures that no overlay messages are propagated to other partitions, while allowing application messages to cross over partitions. This enhances security since all the attacks (Section 2), except message forwarding, rely on overlay messages. For additional assurance, each node can validate a sender ID. For example, if a sender

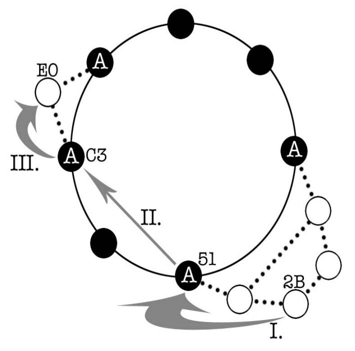

Figure 2. The Stages of Minor-to-Minor Node Phyllo Routing: Phases I, II, and III.

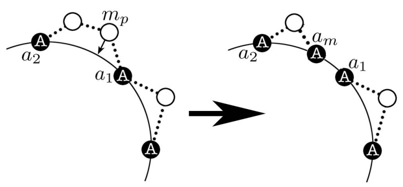

Figure 3. Phyllo Promotion Protocol: Minor node m_p is promoted by a_1 to become a major anchor node, a_m.

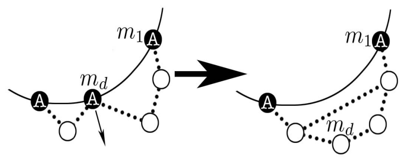

Figure 4. Phyllo Demotion Protocol: Major misbehaving node m_d is demoted by anchor a_1.

is part of a different partition, messages from this sender are discarded since that sender ID is unknown to the recipient.

3.2. Routing

Within each partition, messages are routed following the network's routing protocol. Since Phyllo uses Pastry for its basic routing protocol, every application message is routed towards the node with the ID closest to the message's destination. To enable the partition-based routing, anchor nodes should be able to distinguish the routing in the major partition from the routing in minor partitions. The destination of a message routed in a minor partition should be between the two anchor nodes of that partition; the message is routed in the major partition otherwise.

Some messages, however, require the transition between the major and minor partitions. For instance, the process of routing a message between two minor partitions (Figure 2) can be broken down into three separate stages. First, the message from a minor node is routed in the sender's minor partition and eventually arrives at one of its anchor nodes (Phase I). The anchor node then forwards this message towards the destination (Phase II). The message continues to be routed within the major partition until the message reaches the major node closest to the destination. Finally, if the recipient major node is an anchor node and has minor nodes with IDs closer to that of the destination, the message is routed to the final destination in the minor partition (Phase III). All messages are routed using a combination of these phases.

3.3. Promotion and Demotion Protocols

Phyllo executes the promotion and demotion protocols to move nodes from a minor partition to the major partition, and vice versa. These protocols act as a means to reward or punish nodes for their behavior. The nodes are chosen to be promoted/demoted by the evaluation of a trust metric. We use the trust evaluation rate α to denote the accuracy of this evaluation. For example, $\alpha = 90\%$ indicates that the evaluation algorithm can distinguish good and malicious nodes with 90% accuracy. We assume that Phyllo estimates the trust value of a node accurately (we will investigate the trust evaluation algorithms further as future work). In addition, the promotion protocol is also used to control the size of minor partitions for lessening bottlenecks at the anchor nodes.

The promotion protocol (Figure 3) begins by an anchor node a_1 identifying one of its minor nodes m_p to be promoted. a_1 then contacts m_p as well as the other anchor, a_2, responsible for m_p. Once m_p receives its promotion message, m_p splits its current minor neighbor nodes' information into two new minor neighbor sets – one for nodes between a_1 and m_p, and the other for nodes between m_p and a_2. Meanwhile, a_1 updates its minor partition that contains m_p, removing all nodes not between a_1 and m_p. Similarly, a_2 updates its minor partition. Afterwards, the anchor nodes send their minor partitions updates with regard to the promotion and new partition routing information. a_2 also sends a routing update to the major partition informing of m_p's promotion. Once this procedure completes, m_p becomes a new anchor node a_m and divides the minor partition between a_1 and a_2.

Demotion occurs in the same fashion as promotion. Instead of removing nodes and splitting partitions, the minor partitions are combined. A node m_1 in the major partition targets another major node m_d for demotion (Figure 4). m_1 updates the nodes in the major partition (including m_d) of m_d's demotion. The other major nodes remove m_d from their routing information once the demotion message is received. m_d deletes the major routing information and merges its minor partition routing information together to be new minor routing data. m_d finally notifies all the nodes in the new minor partition of m_d's demotion. When an anchor node is lost, the demotion protocol allows Phyllo to recover by demoting the lost node and updating the partitions' routing information.

What happens if a node disobeys the promotion or demotion protocol? In either case, the disobedient node can only harm itself. Since a node is distinguished as a major or minor node by its peers, Phyllo does not care how a node sees itself. If a node behaves differently from its major/minor designation, all of that node's messages will be dropped because the sender is not known. In the worst-case scenario, the network loses its partitioned topology and functions as the original overlay.

4. Performance Evaluation

In this section, we analyze the performance of our overlay security framework implementation via simulations.

4.1. Implementation and Experimental Setup

We have implemented the techniques described in Section 3 for Phyllo on the Pastry overlay network. For this implementation, we use the two different kinds of promotion protocols: the original promotion protocol (Section 3.3) and a forced promotion as an extension of the original promotion. In the original promotion, the trust value of a minor node should be higher than a configurable threshold to be promoted to the major partition. In contrast, a minor node may be coercively promoted to the major partition in order to split a minor partition. The forced promotion is invoked when the size of the minor partition becomes extremely large (may cause bottlenecks at the anchor nodes). In this case, our implementation promotes the minor node with the highest trust value and with the ID closest to the center of that partition's ID range.

Several methods support the promotion/demotion mechanisms. Whenever a node receives a message, `evaluate(Message)` is invoked. This method allows the node to inspect the contents of the message to determine whether the message has been correctly routed. For promotion and demotion, the node calls `havePromotions()` and `haveDemotions()`, respectively. If either of these returns true, the corresponding node can be retrieved with `getPromotionNode()` and `getDemotionNode()`. Once a node has been promoted

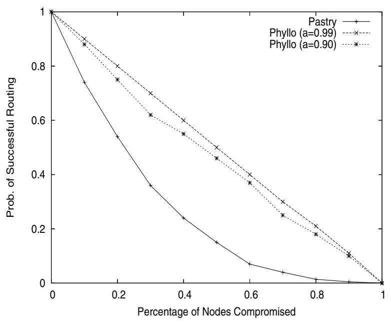

Figure 5. Chances for Successful Overlay Routing

or demoted, `promoted(Node)` and `demoted(Node)` complete the procedures. Note that while Phyllo supports the actual protocols, each overlay network implementation defines the conditions under which a node is promoted or demoted. Finally, `allowJoin(Node)` supports the join process of a new node. A request for join is accepted or rejected based upon some criteria such as partition size or ID distribution.

In our implementation, the minor partition size (including both anchor nodes), R, is strictly bounded by Pastry's leaf set size, L. Whenever $|R| = |L|$, the forced promotion is performed. Otherwise, Phyllo's configurations are equivalent to those of Pastry: ID base 16 ($b = 4$), leaf set size $|L| = 16$, routing table entries $|M| = 16$, and ID space, 2^{128}. Our experiments use overlay network sizes from $N = 100$ to 10,000 and are run on a Dual 2 GHz G5 PowerMac with 1 GB memory and Java 1.4.2.

4.2. Experimental Results

Figure 5 illustrates that Phyllo protects against message forwarding attacks (i.e., compromised nodes do not forward messages) by significantly improving the successful overlay routing ratio. The data show the routing success of 200,000 messages for Pastry and for Phyllo using trust evaluation schemes of $\alpha = 99\%$, and 90% where $|R| = |L| = 16, |M| = 16$ and $N = 5000$. All Phyllo data assume that the minimal number of major nodes is used. Here, the successful overlay routing ratio is the ratio of the number of successful routings to the total number of overlay routings. Pastry has a polynomial routing ratio – the addition of compromised nodes decreases the number of possible message routes by a polynomial factor. Phyllo yields an almost linear relationship, decreasing an attacker's effectiveness. Even with a trust evaluation rate of $\alpha = 90\%$, Phyllo still provides at least a 10-20% improvement over Pastry against forwarding attacks.

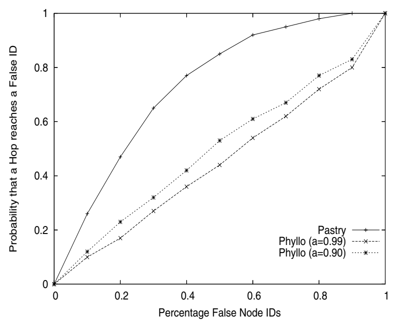

Figure 6. The Probability that a Message is Routed To a False ID

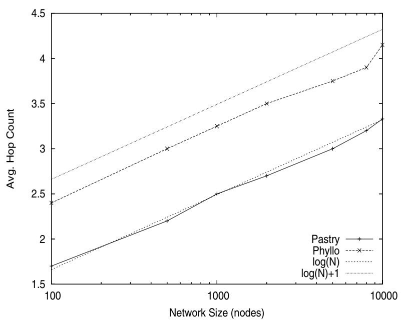

Figure 7. Average Hop Counts for Overlay Routing

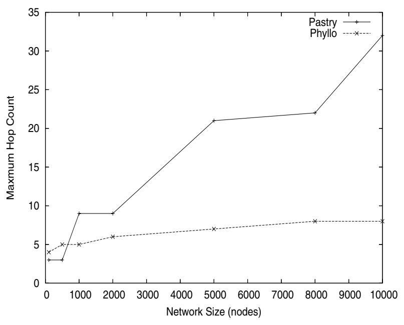

Figure 8. Maximum Number of Routing Hops in Pastry versus Phyllo

Phyllo also mitigates attacks involving bad routing updates (Figure 6). To compare Phyllo's and Pastry's resilience against such attacks, we simulate an attack where the malicious nodes use routing updates to propagate false node IDs throughout the network ($|R| = |L| = 16, |M| = 16$ and $N = 5000$). We assume that a percentage of the total number of IDs are false and the trust evaluation scheme detects these bogus IDs with either 99% or 90% accuracy. 200,000 messages are routed through the network. We track how often a message reaches one of the false IDs. Since this attack resembles the message forwarding attack (except that the compromised nodes have spoofed IDs), the results of these two attacks should be similar. The results show that Phyllo reduces routing attacks by more than a factor of 2 for 20-80% false node IDs.

For network size n, Pastry can route a message between any two nodes in $O(\log_{2^b} n)$ hops [10]. Figure 7 validates that Phyllo is upper-bounded by the same expression. (200,000 messages were sent between random nodes on various sizes of Pastry and Phyllo networks.) Since the major partition of Phyllo is inherently a Pastry ring, Phyllo requires at most as many hops as Pastry from one major node to another major node. To route between two different minor partitions, however, additional hops to route via an anchor node are necessary. Phyllo thus upper-bounds the longest path as the same route in Pastry (from the same source to the same destination) plus two extra hops. Our experiments indicate that the average overall hop count in Phyllo is $O(\log_{2^b} N + 1)$.

Despite having a larger average hop count, Phyllo actually reduces the maximal hop count significantly compared to Pastry. Figure 8 depicts that the maximum hop count of Phyllo is only 8 hops for a network of 10,000 nodes while the maximum hop count of Pastry is at least 32 hops. We analyze 200,000 messages routed from a random source node to a random destination ID when $|R| = |L| = 16, |M| = 16$. As network size increases, the maximum hop count of Pastry increases significantly (from 3 to 32) whereas that of Phyllo increases marginally (from 4 to 8). For a network with unreliable connections (a common problem in the Internet), Phyllo provides more reliable connections with the lower maximum routing distance.

5. Related Work

Security issues in overlay architectures have been studied in several different contexts [2, 11]. These solutions require trusted servers or significant changes to underlying overlay protocols. In addition, none of the solutions can be added to other peer-to-peer systems transparently.

Several overlay systems employ hierarchical architectures for routing, but not for security purposes [4, 15, 5]. Architectures, such as [4], merge multiple overlays into a single peer-to-peer ring, in which a subset of nodes serve as gateways (similar to our anchor nodes) between different overlays. These overlays, however, are still vulnerable to a variety of malicious attacks.

HIERAS [15] divides a P2P ring into several subsections according to lowest link latency. Routing begins at the lowest layers of the hierarchy and then proceeds to the higher layers. While this approach reduces network latency, the system is not protected from malicious nodes dropping or manipulating packets. In addition, neither [4] nor [15] features node transition mechanisms between the hierarchies after joining.

Phyllo bears some resemblance to the Jelly hierarchical framework [5]. Jelly divides nodes into hierarchical subsets, similar to our minor partitions, for load balancing and locality. Nodes in each subset can move from one subset to another in order to balance the hierarchy (similar to our promotion/demotion protocols). Jelly, however, does not incorporate any additional security mechanisms.

Node evaluation has also been an important topic in peer-to-peer security. Some schemes [13, 3] require a new node to contribute resources ("paying its due") prior to active participation. A new node in Phyllo, however, can fully participate in the network without prior resource contribution. Phyllo can also offer the incentive of higher responsibility and marginally better routing performance in exchange of good behavior. Other peer-to-peer overlay trust evaluation algorithms include [1, 6]. Phyllo can adopt such mechanisms, especially for its `evaluate(Message)` method.

Secure Overlay Services (SOS) [7] is a peer-to-peer application that attempts to prevent denial-of-service attacks. Peer-to-Peer Security Layer Framework (P2PSLF) provides encryption methods for secure communication among nodes, access policies, and logging functions for grid networks. P2PSLF assures the confidentiality and integrity of a message; however, P2PSLF does not decrease the chance that a malicious node drops the message.

6. Conclusion and Future Work

We have presented a flexible and configurable security framework, *Phyllo*, that can be implemented on any structured overlay network. The framework mitigates attacks from malicious nodes when these nodes attempt to control message routes, corrupt routing information, and bring down overlay networks. Phyllo includes three essential mechanisms: partitioning, routing, and promotion/demotion. We have implemented a basic version of Phyllo on Pastry and performed experiments in a local machine. Our simulations show that our framework lowers the maximum number of routing hops and exhibits more reliable message delivery in the presence of malicious nodes.

We will conduct larger-scale experiments to evaluate the performance of Phyllo in more realistic environments (e.g., PlanetLab [9]). This includes the implementations of Phyllo on other overlays and their experiments. Analyzing the results from these experiments will allow us to evaluate the quality of the major partition and determine the best minor partition size. This will greatly assist in making recommendations for the major partition with mostly trusted nodes. The best minor partition size should help not overburden anchor nodes, while providing small hop counts and reliable routing. More work will be done to allow Phyllo to work with highly dynamic networks. Currently, we cannot effectively recover from a failed anchor node and have trouble handling nodes joining during a promotion or demotion. We will also explore several different evaluation algorithms to detect potentially harmful nodes for demotion.

References

[1] K. Aberer and Z. Despotovic. Managing Trust in a Peer-2-Peer Information System. In *Proc. of ACM CIKM*, pages 310–317. ACM Press, 2001.

[2] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach. Secure routing for structured peer-to-peer overlay networks. In *Proc. of USENIX OSDI*. ACM Press, 2002.

[3] E. Friedman and P. Resnick. The Social Cost of Cheap Pseudonyms. *Journal of Economics and Management Strategy*, 10(2):173–199, June 2001.

[4] L. Garcés-Erice, E. Biersack, P. Felber, K. Ross, and G. Urvoy-Keller. Hierarchical Peer-to-peer Systems. In *Proc. of IASTED PDCS*. Springer-Verlag, LNCS, 2003.

[5] R. Hsiao and S.-D. Wang. Jelly: a dynamic hierarchical P2P overlay network with load balance and locality. In *Proc. of IEEE ICDCS*, pages 534–540, March 2004.

[6] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The EigenTrust Algorithm for Reputation Management in P2P Networks. In *Proc. of ACM WWW*, pages 640–651, 2003.

[7] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *Proc. of ACM SIGCOMM*, pages 61–72, 2002.

[8] S. J. Nielson, S. A. Crosby, and D. S. Wallach. A Taxonomy of Rational Attacks. In *Proc. of IPTPS*, February 2005.

[9] L. Peterson, T. Anderson, D. Culler, and T. Roscoe. A Blueprint for Introducing Disruptive Technology into the Internet. In *Proc. of the HotNets-I*, October 2002.

[10] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *Proc. of ACM/IFIP/USENIX Middleware*, pages 329–350, Heidelberg, Germany, November 2001.

[11] E. Sit and R. Morris. Security Consideration for Peer-to-Peer Distributed Hash Tables. In *Proc. of IPTPS*, Cambridge, MA, March 2002.

[12] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable peer-to-peer lookup service for Internet applications. In *Proc. of ACM SIGCOMM*, pages 149–160, March 2001.

[13] M. Waldman and D. Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proc. of ACM CCS*, pages 126–135. ACM Press, 2001.

[14] D. Wallach. A Survey of Peer-to-Peer Security Issues. In *Proc. of ACM ISSS*, 2002.

[15] Z. Xu, R. Min, and Y. Hu. HIERAS: a DHT based hierarchical P2P routing algorithm. In *Proc. of IEEE ICPP*, pages 187–194, October 2003.