Design Principles for Low Latency Anonymous Network Systems Secure against Timing Attacks

Rungrat Wiangsripanawan, Willy Susilo and Rei Safavi-Naini

Center for Information Security
School of Information Technology and Computer Science
University of Wollongong, Australia
Email: rw26@uow.edu.au, wsusilo@uow.edu.au, rei@uow.edu.au

Abstract

Low latency anonymous network systems, such as Tor, were considered *secure* against timing attacks when the threat model does not include a global adversary. In this threat model the adversary can only see part of the links in the system. In a recent paper entitled *Low-cost traffic analysis of Tor*, it was shown that a variant of timing attack that does not require a global adversary can be applied to Tor. More importantly, authors claimed that their attack would work on *any* low latency anonymous network systems. The implication of the attack is that *all* low latency anonymous networks will be vulnerable to this attack even if there is no global adversary.

In this paper, we investigate this claim against other low latency anonymous networks, including Tarzan and Morphmix. Our results show that in contrast to the claim of the aforementioned paper, the attack may not be applicable in all cases. Based on our analysis, we draw design principles for secure low latency anonymous network system (also secure against the above attack).
Keywords: Low latency, anonymous, timing attacks, Tor, Tarzan, Morphmix

1 Introduction

Anonymous communication systems were first introduced in the seminal paper of Chaum (Chaum 1981). Conceptually, a message to be anonymized is relayed through a series of nodes called *mix nodes*. Each mix node performs operations that have two main objectives. The first one is to provide *bitwise unlinkability* and is aimed at message content, and the second one is *mixing* that is aimed at message flow. To provide unlinkability messages are padded and encrypted so that the adversary cannot see the content of data packets and so cannot link the content. In each node incoming message is batched and reordered or relayed in a way that is difficult for the adversary to discover its corresponding outgoing message through the message arrival and departure times. Also, to make the attack more difficult, dummy traffic is introduced.

Anonymous communication systems over the Internet can be classified into two categories: systems for high-latency applications and systems for low-latency applications. High latency applications are application that do *not* demand quick responses, such as email systems. On the other hand, low latency applications are applications that *do* need real-time or near real-time responses. Examples include web applications, secure shell (SSH) and instant messenger. Both systems are built based on Chaum's idea. Unlinkability is provided in a similar way in both cases using a sequence of nodes between a sender and its receiver, and using encryption to hide the message content. An intermediate node knows only its predecessor and its successor.

High-latency systems are message-based systems while, low-latency systems are connection-based. That is, for high-latency systems there is one message per one path, and for a new message a new path is created. However, low-latency systems use a path for a period of time and send data as a stream of packets over the same path.

Another difference between the two is due to the time restriction. Anonymous systems for low latency applications may ignore the mixing process that includes batching and reordering, hence, they would be more susceptible to traffic analysis attacks and in particular timing attacks. Timing attacks can be as simple as comparing the difference between the time that packets enter and leave a network, with the time for traversing a route. Timing attacks can be more complex and include extracting traffic patterns of links and comparing them to determine a route. In this paper, we concentrate on the second approach.

The timing attack in (Danezis 2004, Levine, Reiter, Wang & Writght 2004) uses the fact that each node in the network introduces a different delay. The delay can be used to *guess* the correlation between the input links and output links of a node. An adversary can observe links over time and by comparing traffic patterns of all links, determine series of nodes that have similar link patterns and are likely to form a route. Using statistical methods, the adversary can obtain information about the sender and the receiver of a path, or at least *the path* itself. The attack can be avoided by making the timing characteristic of each link *indistinguishable*. This, however, requires an unreasonable amount of mixing operations and cover traffic and hence, long delays. It is not a trivial task to find the right balance between anonymity and delay in these networks.

The aforementioned timing attack assumes a global adversary, who can observe all links in the networks. Under a weaker threat model, i.e. excluding the global adversary, most low latency networks, such as Tor (Dingledine, Mathewson & Syverson 2004) can be considered *secure*. In this weaker threat model, the adversary is allowed to only observe a fraction of the links. This is a plausible assumption considering the systems are operated over public networks, such as the Internet and so there is no assurance that the timing attacks can achieve more than finding parts of the routes with non-negligible success probability.

Recently, Murdoch and Danezis have shown a low cost attack that can successfully "break" low latency systems, and does not use a global adversary (Murdoch & Danezis 2005). Their attack is based on a traffic analysis attack proposed by Danezis (Danezis 2004). In their attack, Tor's provided anonymity can be broken by an attacker that only has a partial view of the network or is one of the Tor nodes. The attack works because Tor removes the mixing operation that has been used in its earlier version, and instead processes its input queues in a round robin fashion. The Tor node is responsible for receiving and forwarding each stream's packets. A corrupted Tor node can create connections to other Tor nodes and so indirectly estimate other nodes' traffic volumes at each time. These estimates use the difference in the latency of streams that are sent and received back using those connections. As the traffic volume or traffic load on each Tor node is a result of the traffic load of all relayed connections on that Tor node, the technique in (Danezis 2004) can be used to estimate the traffic pattern of each node and utlimately a good estimate of the route. Authors noted that their scheme can be applied to *any* anonymous low latency systems. This significantly invalidates the threat model used in many low latency anonymous systems.

Our Contributions
We examine the attack proposed by Murdoch and Danezis (Murdoch & Danezis 2005), and investigate if it works for other low latency anonymity networks. We note that Tarzan (Freedman & Morris 2002) and MorphMix (Rennhard & Plattner 2002) work differently from Tor. In particular, they both employ peer-to-peer architecture, whereas Tor tends to rely on dedicate servers. Also, Tarzan includes some mixing operations and cover traffic, which does not exist in Tor. Moreover, MorphMix allows an intermediate node to select part of the anonymous path, whereas in Tor, a Tor client chooses the whole path by himself.

Our analysis has two directions. Firstly, we focus on the *latency* of the system, to verify whether it can be used to represent the traffic load of each node. Secondly, we look at the effectiveness of the attack on different architectures and mechanisms. We will use our analysis to provide design principles for building secure low latency anonymity systems that do not suffer from these types of attacks.

Organization of The Paper
The rest of this paper is organized as follows. In Section 2, we briefly describe the related works in this area. In particular, we review the three existing anonymous network systems, namely Tor, Tarzan and Morphmix and highlight the differences among them. In Section 3, we review the low cost attack on Tor as described by Murdoch and Danezis (Murdoch & Danezis 2005). We note that this attack is claimed to be applicable in *any* low latency network systems. In contrast to this claim, we provide a counter example in Section 4 where the attack fails to be conducted. In particular, we discuss the possibility of employing this attack in Morphmix, and we show that the result does not provide the required information as claimed. In Section 5, we derive some necessary conditions required when building a secure low latency network system. By adhering these necessary conditions, a secure low latency network system can be built and the resulting system will not suffer from the low cost attacks. We also provide some related discussions to highlight our results. Finally, Section 6 concludes the paper.

2 Related Works

In this section, we will briefly review the three existing anonymous network systems, namely Tor, Tarzan amd Morphmix.

2.1 Tor

Tor, the second-generation Onion Routing, is a circuit-based low-latency anonymous communication service (Dingledine et al. 2004). It is an improved version of the *Onion Routing.* Onion Routing(OR) is an overlay system that aims to provide anonymous communication to applications such as web browsing, instant messenger and secure shell. As the OR's designs have several flaws and limitations when being deployed, Tor has included several additional features that OR does not provide. Some of them are perfect forward secrecy, congestion control, directory services, integrity checking, configurable exit policies, and rendezvous point and hidden service. Tor also removes features that are considered by its authors as being unnecessary. These features are mixing, padding and traffic shaping.

There are three entities: a Tor client, Tor servers (Tor nodes), and a recipient. Logically, a Tor client is a sender that wants to have an anonymous communication with its recipient. It is an *Onion Proxy* in OR. Tor servers are intermediate nodes or *Onion Routers* in OR. They are responsible for routing streams to its next nodes in accordance to what the Tor client instructs them. Like OR, Tor calls the last Tor node, before a recipient, *the exit node.* The recipient does not need to be a member of the Tor network. That is because the exit node acts as a guardian between the open world (recipients) and the Tor network.

Similar to OR, a Tor client selects a number of Tor servers as members of a circuit (OR and Tor call a path *a circuit*). Where OR restricts one circuit for one TCP stream, Tor allows many TCP streams to share the circuit. Circuits are constructed *a priori.* The main responsibility of the Tor client is to set up a circuit and establish common keys between the client and all intermediate nodes. These keys are used later when the client wants to send relay cells to its recipient and vice versa. Tor fixes a circuit's size to be 3 Tor server nodes. More details of circuit construction can be found in (Dingledine et al. 2004).

When the client wants to send data to its recipient anonymously, for example, when a user starts browsing a website, streams of packets are divided into fixed-sized cells. The Tor's cell size is 512 bytes. Then, they are wrapped layer by layer using session keys derived from pre-negotiated common keys. This is done in a way that when they are unwrapped by the Tor server, the node is able to know merely its predecessor and successor nodes. Unlike OR that provides mixing processes, the incoming cells to the Tor node are simply put into queues, processed and sent out in a first come first serve fashion.

Tor Threats Model
An adversary's goal is to observe both the initiator and the recipient. Like all other practical low latency anonymous systems, Tor does not provide any mechanisms to protect against a global passive adversary. However, it is designed to prevent the system against an adversary that has the following capacities:

- the adversary who can observe some fraction of network traffic;
- the adversary who can generate, modify, delete or delay traffic;
- the adversary who can operate onion routers of his own;

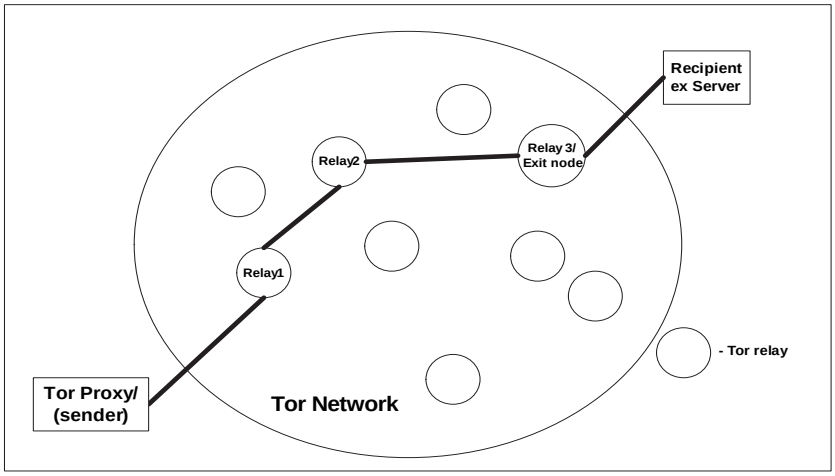

Figure 1: Tor architecture

- the adversary who can compromise some fractions of onion routers.

Attacks using the network traffic can be classified into two main categories: *traffic confirmation attacks* and *traffic analysis attacks*. Each category consists of both passive and active attacks. Traffic confirmation attacks are attacks where the adversary uses a traffic pattern to confirm his guess. For example, the adversary suspects that Alice is talking to Bob. Passively, he observes traffic at both Alice and Bob's ends and uses a pattern that obtained from timing or volume of packets that enter and leave both ends to verify his suspicion; or actively embeds timing signatures into the traffic between these two nodes to force the distinct patterns that can be recognized. However, traffic analysis attacks are attacks that the adversary learns which points in the network he should attack by using traffic patterns. For example, the passive adversary can observe the network edges and then uses relationships in timing or volume of packets to correlate traffic that enter or leave the network; or the active adversary can insert a pattern into traffic that can be detectable afterward.

Examples of the traffic analysis attacks that could be mounted with Tor are as follows. The passive adversary can use other externally visible user-selected options such as timing of packets to correlate traffics. The active adversary can replay traffic; or he can choose to deny service to trustworthy routers and move them to compromised router; or he can deny services to users and see if traffic stops elsewhere.

Tor and Traffic analysis
Tor considers traffic confirmation attacks, such as end-to-end timing correlation, outside their design goal. Tor designers focus their threat model merely on the traffic analysis attacks (Dingledine et al. 2004). Since Tor does not include the global passive or active adversary in its threat model, some traffic analysis attacks, such as observing traffic pattern, can be discarded. Dingledine and Mathewson (Dingledine et al. 2004) provide more details on how Tor defends other traffic analysis attacks. There are also other aspects of attacks that are harmful to Tor, namely attacks with directory service and rendezvous points. However, they are outside the scope of this paper. More details can also be found in (Dingledine et al. 2004).

2.2 Tarzan

Tarzan is another low latency anonymous system, which is also based on the Chaum's mix concept. Similar to other low latency anonymous systems, it is aimed to provide anonymity for applications such as web-application or instant messenger. Unlike Tor, Tarzan is based on peer-to-peer architecture. Each Tarzan node can be both a Tarzan client (the sender) or a Tarzan relay. This is done to avoid end-to-end timing analysis of an entry and exit nodes. That is, any one can join and leave the network and all nodes can be potential initiators. Tarzan provides peer discovery by using a protocol based on the gossip-based mechanism similar to the one in (Harchol-Balter, Leighton & Lewin 1999). Due to the peer-to-peer characteristic, an adversary can imitate himself as many Tarzan nodes as he wishes. Therefore, Tarzan is equipped with a mechanism to store peers in a way that reduces a chance that a malicious node is selected. This is done by hashing the node's IP address according to its subnet and categorizing its result into levels.

The recipient in Tarzan does not necessarily have to be a Tarzan node; it can be any node in the network. Tarzan provides a mechanism to get through it through *PNAT*. Tarzan claims that it is resistant to the global adversary's attack, that is achieved via its cover traffic mechanism, known as *mimic traffic*. To prevent an overwhelming network consumption, Tarzan limits the mimic traffic of each Tarzan node to merely with some of its peers.

To illustrate this, when joining the network, after discovering all other peers, the Tarzan node selects a number of nodes as its mimics. Then, when an anonymous connection is required, the next relay node must be chosen from the node's mimic list.

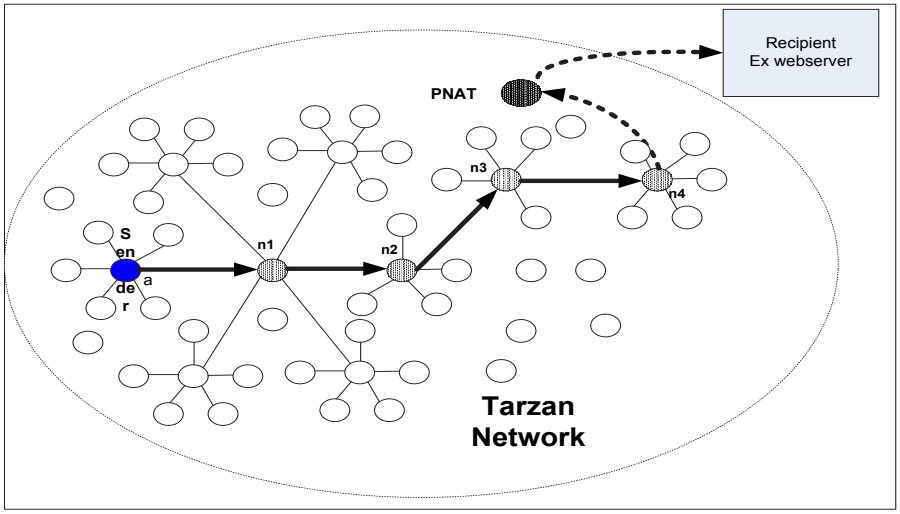

Figure 2: Tarzan Architecture with mimics

When a Tarzan node a wants to have an anonymous connection with its recipient such as a web server svr, assuming that the anonymous tunnel has $l+1$ length where l is a number of the nodes in the tunnel, a selects its first hop from its mimic list, say $n1$. Then, a asks $n1$ for $n1$'s mimic list (l_{n1}). a chooses the 2^{nd} hop, from (l_{n1}). Then, a repeats this process until l hops. Finally, a chooses the last node randomly from a's peer database. This last node acts as an exit node in Tor but Tarzan names it $PNAT$. Therefore, the connections has the following path: $a \rightarrow n_1 \rightarrow n_2 \rightarrow ... \rightarrow n_l \rightarrow PNAT \rightarrow svr$. It is important to note that $PNAT$ is selected randomly from all peers in the database not from n_l's mimic list, otherwise numbers of available paths are limited. Figure 2 illustrates Tarzan network's architecture and its tunnel connections. In this example, each Tarzan node has approximately 6 mimics.

2.3 MorphMix

MorphMix (Rennhard & Plattner 2002, Rennhard & Plattner 2004) is another low latency anonymous system. Its main objective is to provide a practical anonymous communication to the masses. It is based on a peer-to-peer architecture. Similar to Tor, MorphMix is a circuit-based mix system that makes use of fix-length cells and layered encryption to establish a circuit through other nodes. Cover traffic is removed unless really required. The circuit, the first node, the last node and the nodes in between are called the *anonymous tunnel*, the *initiator*, the *final node* and the *intermediate* nodes, respectively. Unlike Tor or Tarzan, the intermediate nodes in MorphMix are not entirely chosen by the initiator. Rather, MorphMix allows each intermediate node to select its successor. When an initiator node a wants to establish an anonymous tunnel, it selects the first intermediate nodes b from its current neighbor list. Then, it establishes a symmetric key between them. The shared key is used for layered encryption. To extend the tunnel, a asks b for a selection of nodes that should be used as its next hop. b sends a a set of its recommended nodes chosen from b's neighbors. a then chooses one of them, for example, node c. a appends c to b in the tunnel. A symmetric key between a and c is created and then, sends to c through b. To prevent b from being man-in-the-middle attack, MorphMix introduces a *witness*. The witness's duty is to act as a third party during the next hop selection process. It allows the initiator a to establish a shared key with the appended node c through b, without revealing c's key information to b or without b being a malicious node. Figure 3 is cited from (Rennhard & Plattner 2002). It shows how MorphMix selects the next hop with the witness's help. Assuming that the connection between the initiator a and b has already been established. The complete procedure is illustrated as follows.

1. a selects a witness w from the set of nodes it already knows. It then generates its half of the key information DH_a and encrypts it with W's public key together with the nonce($\{nonce_1, DH_a\}_{PuKw}$). Note that a nonce is used against replay attack and s is a number of nodes a wants b to offer. *Due to DH_a is encrypted with the w's public key. b has no idea what this information is.*

2. After b receives the message, it sends the encrypted DH_a ($\{nonce_1, DH_a\}_{PuKw}$) to w with its selection of nodes and their public keys ($\{ip_c, PuK_c, ip_d, PuK_d, ip_e, PuK_e\}$).

3. w has two tasks. First, it decrypts $\{nonce_1, DH_a\}_{PuKw}$ to get DH_a. Then, w selects randomly node c as the next hop. Next, it sends request for next hop to c. This includes b's information $\{ip_b, PuK_b\}$ together with a'secret to c.

4. c checks if it will accept the request. If so, it sends ok-message back to w.

5. w signs a set of nodes that b selects together with $nonce_1$ and sends back to b with its chosen node for the next hop put first after the $nonce_1$ in the signature. The signature is used as a receipt from w to a.

6. b receives the message from w. It knows that its next hop is c. It then generates an identifier(id) that is used to identify the anonymous tunnel of this link between b and c. Then, it sends this id with the new nonce ($nonce_2$) to c.

7. c replies b with its DH-exchange part with id.

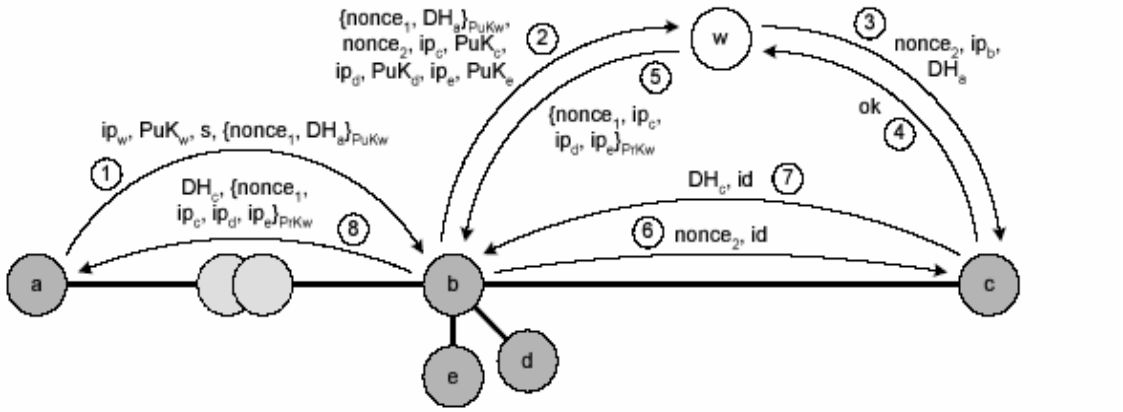

Figure 3: MorphMix hop selections.

8. *b* sends to *a* *c*'s DH part and the signature of *w* that consists of list of chosen nodes.

Unlike any other systems, MorphMix does not require that any MorphMix node must have knowledge of all other nodes. Rather, MorphMix pays more attention to collusion's detection of malicious nodes.

3 A Low Cost Attack in Low Latency Anonymous System

In this section, we review a low cost attack on Tor as described in (Murdoch & Danezis 2005). The term *low cost* comes from the fact that the adversary requires only a partial view of the network, i.e. being one of the Tor nodes, to attack the Tor network. The attack shows that the Tor system is vulnerable to a variant of timing analysis attack, even though this attack was claimed to be *prevented* in the original Tor threats model. That is, the claim that traffic analysis attacks on Tor cannot be done without a global passive adversary is indeed false.

The attack, conceptually, takes advantage of a seemingly unavoidable limitation of the low latency anonymous system; that is *time*. The goal of the attack is to infer which nodes are being used to relay streams in the Tor circuit. This greatly decreases the anonymity properties of Tor system. Experimental results are conducted and presented to support the theoretical attack. In the end, the authors further claim that this attack should be applicable to *any* low latency anonymous system, such as Tarzan and MorphMix.

The Idea of The Attack

The idea comes from the known fact that delay cannot be induced much in the low latency systems. Hence, the timing pattern of packets should persist throughout a circuit. Tor had survived the typical timing attack because its designers believed that the global passive adversary is hardly possible. It does not consider this type of attacks in their threats model. The *low cost* attack indeed disproves this argument and shows that the initial design is still vulnerable to the timing attacks' variants. It is true that the adversary cannot observe all links in the network and mount attacks as described in (Danezis 2004). Nevertheless, being one of the Tor nodes does not prohibit the adversary to measure latencies between all the nodes and itself. In particular, these latencies can be used to infer to traffic volumes of the nodes that it is communicating. Since the traffic pattern of each node can be obtained from its traffic volume, the adversary can use the techniques described in (Danezis 2004) to do further analysis. Then, he is able to infer which nodes carry the similar traffic pattern. In other words, they are the relay nodes in the same circuit.

The idea is supported by the Tor architecture. The Tor node has given each connection its own buffer and processes these buffers in a round robin fashion. When the buffer has no stream, it simply ignores and moves to process the next buffer. More importantly, the mixing process has been *removed* from Tor due to its designers' doubts on capacity and practicality (Dingledine et al. 2004). Hence, when a new connection is established; or when any existing connection is removed; or when the traffics of the existing connections are changed, the traffic loads on the Tor node is changed. This affects the responses that this Tor node would have on connections with other Tor nodes that are previously established and are still currently working. Consequently, due to this same reasons, the traffic loads of the other Tor nodes are changed as well. Therefore, they can conclude that the change of the traffic load on one Tor node affects other Tor nodes that have connections to it. Hence, for nodes that are on the same circuit, their traffic loads should result in the same pattern of effects. Note that the change of traffic load occurred from the Tor node's environments, such as its CPU load, is ignored.

Entities Involved

The adversary merely requires to be a member of the Tor's system. That is, being one of the Tor clients.

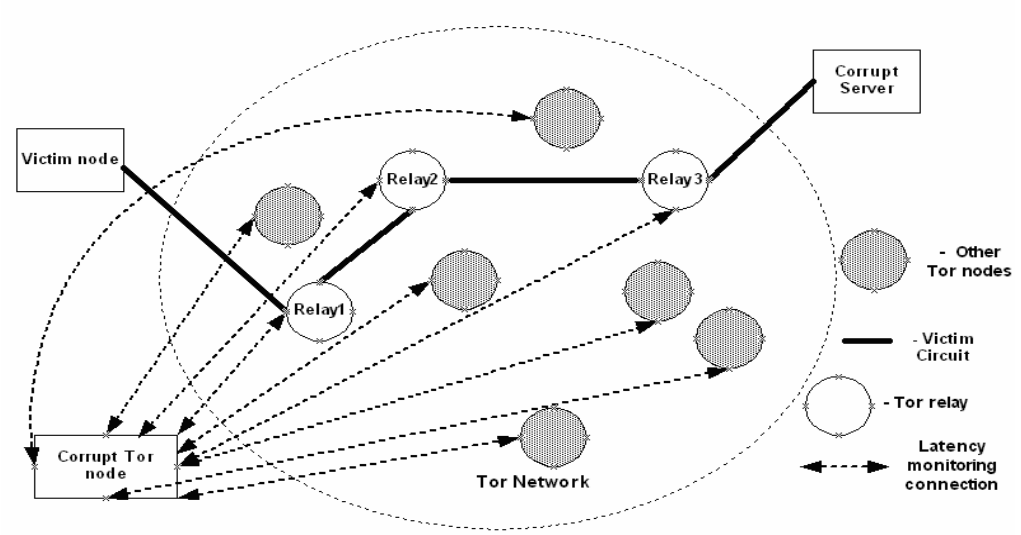

Figure 4: Low Cost Attack Model in Tor

This node is called *a corrupt node* or *a probe node*.

Attack Model
Conceptually, the attack works as follows.

- A corrupt Tor node establishes connections to other Tor nodes in order to measure these connections' latencies.
- The corrupt Tor node keeps monitoring latencies of all these connections during a reasonable time period.
- The latencies values are used to estimate traffic loads of the Tor nodes' that the corrupt node makes connection with.
- Traffic patterns are derived from the traffic loads.
- When the adversary has the traffic pattern of all nodes, it can further mount an attack similar to ones in (Danezis 2004, Levine et al. 2004).

To make their attack more powerful, a variant of the attack is proposed. That is, a network server to which a user is connected to, is corrupted. Thus, there is no need to observe a connection to extract the traffic pattern. The adversary can choose a traffic pattern that is easily detected and sends its streams through the corrupt server. The aim of the attack is to find a path between a victim node and the corrupt server. This greatly reduces the anonymity of the system to be at the same level with a simple proxy. Finally, it is claimed that the attack would work with other low latency network systems, including Tarzan and MorphMix.

In the next section, we will investigate this argument and show that the claim made in this paper is only valid under some restricted conditions. Figure 4 and figure 5 illustrate the low cost attack model and its procedure respectively.

4 Analysis of the Low Cost Timing Attack against Tarzan and Morphmix

To perform this attack with other systems, we employ the same model as the one used by Tor. The adversary requires at least two entities: a *corrupt node* and a *corrupt server*. A corrupt node is changed to a sender node in each particular system. That is, a Tarzan client in Tarzan, and an initiator node in MorphMix.

Next, a corrupt node needs to acquire a list of all other nodes in the system. Then, it establishes connections to these nodes so that it can monitor their connections' latencies. Connections are monitored for a period of time. During that time, the corrupt server keeps sending its traffic into the system. When the monitoring period finishes, latency of each connection is used to estimate the traffic load of that particular node. Then, the traffic load is compared with the server traffic. If it results in similarity, the node is concluded as one of the intermediate nodes in the path. When traffic of all nodes are compared, the possible path(s) is/are derived.

Thus, the attack would be successful under the following three conditions:

- Latencies received at the corrupt node indeed represent traffic loads of the target nodes.
- The corrupt node must be able to know other participants in the network.
- The corrupt node must be able to establish a *direct* connection to *all* nodes it wants to monitor.

The attack is successful in Tor because the Tor architecture satisfies these conditions. Firstly, due to the fact that Tor removes mixing operations and cover traffic, its timing characteristic is retained. This is supported by the experiment result in (Murdoch & Danezis 2005). Secondly, Tor provides a directory service that a Tor client can acquire a list of all other Tor servers. Third, there is no restriction to prohibit

At a corrupt node	At a corrupt server
Preparation	
Find a list of all other nodes ($\{1, 2, ..., N\}$)	Prepare target stream ($S(t)$)
Action	
1. for $i = 1$ to N make connection to each $node_i$;	
2. for $i = 1$ to N	
2.1 while t record latency of each $node_i$ ($L(i)$);	send $S(t)$
2.2 derive $\rightarrow T(i)$ traffic load of $node_i$	
2.3 compare $T(i)$ with the server traffic $S(t)$	
2.4 if $T(i) \approx S(t)$ then $node_i$ is a relay in the path.	
3. Obtain a path, for example, $node_1 \rightarrow node_3 \rightarrow node_6$	

Figure 5: Low cost attack's procedure.

the Tor client not to establish a connection to all Tor servers.

4.1 Low Cost Attack in Tarzan

To investigate if the attack is applicable with Tarzan, we will analyze what influences each condition has provided to the system.

Latency
There are two differences between Tarzan and Tor that should affect the latency. The first one is Tor operates its queue in a round-robin fashion. Secondly, Tor does not include mixing operations and cover traffic. The case is different for Tarzan. Tarzan provides a mimic mechanism. That is, even though Tarzan does not mention how each Tarzan node manages its incoming queues, Tarzan controls the rate of output links according to the average rates of its input links. When each link does not reach the rate it requires, Tarzan inserts dummy data. Also, prior to sending out its streams to output links, the Tarzan node does some mixing and batching within each link's outgoing queue.

Then, the question is if this so-called *mimic* has enough influence to destroy the timing characteristic of each traffic stream. At this stage, it seems that the only way to prove this statement is to conduct an experiment in the same fashion as the experiment conducted in Tor. Unfortunately, Tarzan does not provide the test-bed that includes the mimic part. We leave this as an open research question. For the purpose of this analysis, we would treat the Tarzan node operations as a black box and assume that timing signature is not destroyed. Hence, the comparison between the attack on Tor and Tarzan is still fair.

Node discovering ability
A Tarzan node discovers other peers through a mechanism based on a gossip-protocol. Thus, each Tarzan node can gather all information about all peers. What seems to be a problem is Tarzan is based on peer-to-peer architecture so that the number of nodes should be huge and the network should be dynamic. Then, whether it is possible that the corrupt node can monitor all others' node latency is dubious. However, since we are merely concerned with the ability to know other peers in the network, Tarzan is still considered to satisfy this condition.

Connection Establishment ability
After recognizing all other nodes in the network, if the corrupt node is able to establish a direct connection with all nodes through an anonymous tunnel in Tarzan network, then this will satisfy the third requirement. Unlike Tor, Tarzan restricts its traffic through merely its mimics. Therefore, when target nodes are not mimics of a corrupt node, the corrupt Tarzan node may not be able to make a one-hop connection to each target node. When the connections require more than one hop, the corrupt Tarzan node cannot be sure about the correctness of the latency it measures. Then, the traffic load may be interfered by the other traffic load of the nodes in between the connections. Up to this point, it seems that the low latency attack may not work in Tarzan. Nevertheless, the adversary is fortunate. Tarzan employs $PNAT$, which is the last node in the tunnel prior to the recipient. Unlike other intermediate nodes in the tunnel that their successors and their predecessors are restricted to their mimics, the $PNAT$ can be any node in Tarzan network. Therefore, to measure latency of other node in the network with a single hop connection, the corrupt node can treat each target node as the $PNAT$ of that connection. Then, there will be no problem with the mimic traffic.

In summary, the low cost attacks should be applicable with the Tarzan architecture, unless Tarzan's mimic traffic can destroy timing characteristic of the streams or the $PNAT$ mechanism is modified to require the exit node to be part of the mimics.

4.2 Low Cost Attack in MorphMix

The major differences between MorphMix and Tor seems to be the architectures of the systems and the method to select tunnels' members and the exit node. MorphMix works in a peer-to-peer environment where Tor has dedicate servers acting as Mix nodes. In Tor, a Tor client is the one who selects all participants in its anonymous tunnel by itself. However, a MorphMix node selects its participants through suggestions of each intermediate node in the tunnel. Tor does have an exit node but MorphMix does not. We conduct our investigation whether the low cost attack will be applicable to MorphMix based on the same three criteria mentioned earlier.

Latency
MorphMix nodes appear to work in the similar fashion as Tor nodes, after tunnels are established. That is, no cover traffic mechanism is included. Thus, in this aspect, the attack should be applicable to MorphMix.

Node discovering ability
There is no requirement in MorphMix that each MorphMix node must have knowledge of *all* other MorphMix nodes in the system. Simply, each node requires a handful of other nodes obtained locally such as from its neighbors. Then, when it wants to establish a tunnel, it continually asks each hop one at a time to recommend a set of possible next hop. This allows nodes in MorphMix system to create anonymous tunnels without concerning the knowledge of all nodes in the system.

The implication is as follows. When the low cost attack is employed, the corrupt MorphMix node has a problem with acquiring a complete list of all running MorphMix nodes. That is, it must discover all nodes first. This is not a trivial exercise, in particular, when the discovery can only be done through MorphMix tunnel establishment mechanism. Therefore, an effective searching algorithm is required. Nevertheless, this involves a lots of work. Hence, the so-called "low" cost attack will now become "costly". Also, by the time all nodes are discovered, the list may be outdated, since some peers may have disappeared from the network. Moreover, due to loose knowledge of other nodes' presences, there is no assurance that all

nodes in the MorphMix network are connected. In this case, the attack cannot even be conducted.

Connection Establishment ability
There is no restriction in MorphMix connection. Hence, each MorphMix node can have a one-hop connection to other MorphMix node directly.

In short, the current attack is *not* applicable to the MorphMix architecture because the corrupt node is lacking the knowledge of a complete list of all MorphMix nodes.

5 Design Requirements for Secure Low Latency Networks

There are two essential conditions required in mounting the low-cost attack. Firstly, the knowledge of all other nodes in the system by a corrupt node is essential. Without this knowledge, this type of attacks cannot be performed. Secondly, the latency being probed must genuinely represent the traffic load.

Based on these observations, we derive the following conclusions. There are two alternative approaches that can be taken to prevent the low cost traffic analysis attacks.

1. Preventing all nodes from gathering information of the whole network , i.e. a list of all nodes. This implies that we only need to provide adequate information so that an anonymous tunnel can be created.

2. Inserting cover traffic into streams in the network in a way such that the network cannot find the streams' signature.

Based on either of these two possible approaches, we can build a secure low latency network that will not be susceptible against the low cost attacks.

It is important to note that all strategies suggested by (Murdoch & Danezis 2005) fall into the second approach. However, as they introduce more latency to each connection and involve a covert channel, there remains an open problem in what degree that cover traffic should be employed. Hence, the first alternative sheds a new light in preventing this attack.

Further Discussions
The significant part for anonymity preserving in a low latency anonymous system appears to be in the second scenario, that is, the message flow or traffic flow. This is due to the main restriction of the system, which is an intolerable long delay. When enough delay is introduced, it distorts the relay node capability to mix or make its incoming and outgoing streams indistinguishable. This leaves some "clue" for the adversary to finally break the anonymity of the system. However, having multiple nodes in the anonymous tunnel hardens the adversary in the sense that he must be able to control *all* nodes or links. This group of adversary is called global adversaries.

It seems rather difficult to provide anonymous low latency systems that are both resistant to global adversaries and provide acceptable delay for applications. Fortunately, it is not a trivial task to be the global adversary in the Internet as the Internet hosts are distributed around the globe under different domains and authorities. Therefore, it is considered to be reasonable to assume that a system is *secure* under a weaker type of adversaries. These adversaries are allowed to do several things both passively and actively, such as observing some fractions of network traffic or generating, modifying, deleting or delaying traffic or operating some intermediate nodes; except the requirement for observing *all* links.

Systems, such as Tor, claim that they provide enough countermeasure against this type of attacks. However, they have not considered that a variant of the attack can be applied without having to involve a global adversary. By taking advantage over the timing constraint and the removing of cover traffic, the low cost attack has successfully attacked the weaker threat model in Tor, and therefore, this model is regarded as insufficient.

According to our investigation with other systems, which are Tarzan and Morphmix, the attack is considered to be valid to *all* low latency networks with the requirement that the system must allow a node to obtain a list of all other nodes in the network. Otherwise, if this capability is prohibited, the weaker threat model is acceptable.

Hence, for any anonymous low latency system that wants to claim the weaker threat model that secures against a timing attack, this condition must be enforced. Under this condition, each node is only allowed to know a subset of neighbors, but not *all* of them.

This can be used as another supportive reason to favor peer-to-peer based systems over dedicated-server based systems in designing a low latency anonymous network. This is due to the fact that a number of nodes in the peer-to-peer system is huge. Thus, if an adversary manages to find all the nodes, the list would be outdated as it is hard to obtain the list of all nodes within a short time.

6 Conclusion

In this paper, we investigated one of the attacks in low latency anonymous network systems, namely the low cost traffic analysis attack. This attack is an important one, since it has proven to be successful in attacking a system like Tor, which was believed to be secure under the *weaker* threat model. Moreover, these types of attacks are claimed to work with *any* low latency anonymous systems. We presented a case where this attack is not applicable. We also investigate some important properties that need to be ensured whilst building low latency network systems, so that they will not be susceptible against these types of attacks. Hence, we provided some necessary conditions that are important and need to be adhered when designing the low latency anonymous networks. We note that cautions must be exercised upon building this type of networks.

References

Chaum, D. (1981), Untraceable electronic mail, return address, and digital pseudonyms, *in* 'Communiation of ACM', Vol. 24, pp. 84–88.

Danezis, G. (2004), The traffic analysis of continuous-time mixes, *in* 'Proceedings of Privacy Enhancing Technologies workshop (PET2004)', Vol. 3424 of *LNCS*.

Dingledine, R., Mathewson, N. & Syverson, P. (2004), Tor: the second-generation onion router, *in* 'Proceedings of the 13th USENIX Security Symposium. San Diego, CA, USA. 9-13 Aug. 2004.'.

Freedman, M. J. & Morris, R. (2002), Tarzan: A peer-to-peer anonymizing network layer, *in* 'CCS'02 , Washington, DC, USA', ACM.

Harchol-Balter, M., Leighton, T. & Lewin, D. (1999), Resource discovery in distributed networks, *in*

'PODC '99: Proceedings of the eighteenth annual ACM symposium on Principles of distributed computing', ACM Press, New York, NY, USA, pp. 229–237.

Levine, B. N., Reiter, M. K., Wang, C. & Wrightt, M. K. (2004), Timing attacks in low-latency mix-based systems, *in* 'Proceedings of Financial Cryptograhy (FC'04)'.

Murdoch, S. J. & Danezis, G. (2005), Low cost traffic analysis of Tor, *in* 'IEEE Symposium on Security and Privacy, Oakland, California, USA, May 8 11, 2005'.

Rennhard, M. & Plattner, B. (2002), Introducing Morphmix: peer-to-peer based anonymous internet usage with collusion detection, *in* 'WPES '02: Proceedings of the 2002 ACM workshop on Privacy in the Electronic Society', ACM Press, New York, NY, USA, pp. 91–102.

Rennhard, M. & Plattner, B. (2004), Practical anonymity for the masses with Morphmix, *in* 'Financial Cryptography. 8th International Conference, FC 2004. Revised Papers.'.