Privacy preserving lookups with In-DHT Onion Routing [RFC]

DRAFT: This document is work in progress. Please send your comments, suggestions and corrections to gpestana@hashmatter.com or join the conversation at ((gpestana), n.d.)

Abstract: Distributed Hash Tables (DHTs) are.. A knowns privacy vulnerability of DHTs protocols are.. Onion routing is.. In this paper, we consider several details and open questions for using onion routing to protect the privacy lookup initiators in DHTs in a scalable and secure manner.

Introduction

Distributed hash tables

DHT is a network protocol that implements a hash table over a P2P network. The DHT participants have access to an API `store(data, ID)` and `lookup(ID)`, which allows peers to store data in the network and find it. Each node in the network is assigned with an unique ID. In addition, the data stored in the network also has an ID which overlaps the node ID domain. When data is stored (i.e. `store(data, ID)` is issued by a peer), it is routed between peers until it reaches the peer with the closest ID. This design enables peers to coordinate itself to store and retrieve keyed values without central point of authority or registry and provide a decentralized.

Different flavors of DHT protocols implement the routing mechanism - how to pass request messages to the correct peers in the network, so that the content is found - and overlay network topology - what network organization the network adopts - in different ways. The network topology is shaped based on which network addresses a peers has access to.

The problem: lookup initiator privacy

When peers make a network request for a specific data ID, usually a chain of network requests starts in order for the lookup initiator to learn which peer in the network has the data - or if the data is not available. While the open collaborative nature of DHTs enables large networks of peers to resolve requests without a a single point of failure and central authority, it also leaks information to anyone in the network regarding what peers are requesting. Potentially any peer in the network has access to which data is requested by any network peer. On applications where sensitive data is stored and requested, this poses a privacy problem to every peer in the network and renders the whole network unusable from a privacy perspective.

A (potential) solution: In-DHT onion routing

In-DHT onion routing consist of encrypting the lookup request in multiple layers, where each layer can be decrypted by only one network peer - called relay. After decrypting one layer of the packet, the relay gains access to the information necessary to route the remaining encrypted packet to the next relay or to perform the lookup if all the layers have been decrypted already. The exit relay - the last peer in the onion circuit responsible for making the request - also has access to a response packet which consists of the encrypted onion packet that must be used for the response to be sent back to the original lookup initiator.

The first step for a peer to use in-dht onion routing is to construct an onion circuit, An onion circuit is the set of peers that will be as relays and will be decrypting and forwarding the onion packet in the network. Tor uses a central authority directory with information necessary to construct the onion circuit and

Threat model

Onion routing security breaks against an adversary that can observe the whole network and . Onion routing is designed to be secure against adversaries with only partial view of the network, so we do not consider an attacker that have a full view of the network at any time and can see all links between peers (or, as (Syverson 2013) refers to it, “The Man”).

We consider a local adversary with partial view of the network and that controls up to a fraction `f` of colliding nodes in the network. We consider `f` to be `0.2`.

Goals

In the current work, we define as main goal to design a DHT which leverages onion routing to achieve the following goals:

- Provide anonymity and message unlinkability to lookup initiators;
- Secure against local adversary with partial view of the network that can control up to 20% of the network (see threat model above);
- Network information management infrastructure (e.g. peer relay directories, PKI infrastructure, etc) are completely decentralized;
- Low latency network;
- Relatively low computational and time overhead;

In-DHT onion routing

Protocol

Onion routing vulnerabilities

Previous research have demonstrated that onion routing is vulnerable to a set of de-anonymizing attacks under the assumption that an active and passive attacker with partial view of the network - and in some cases even weaker. These attacks are effective given the low-latency nature of the protocol. The In-DHT onion routing is vulnerable to the same set of attacks, which are:

- **Adversary controls entry and exit relay**: this timing attack is effective regardless of the entropy in the network (Syverson 2013). An attacker which controls entry and exit relay is able to correlate and de-anonymize packets in the circuit, regardless of the number of peers using the same relays.
- …

Note that low latency anonymity networks - such are onion routing-based netowkrs - are fundamentally broken against “The Man” (Syverson 2013). They do offer, though, some protection against weaker adversaries and may be an interesting trade-off between latency, overhead and anonymity for the DHT use case.

Provably secure onion circuit building

Conceptually, a lookup initiator must be able to hide from internal and external peers which nodes were selected when constructing the onion circuit. Formally that is translated by saying that the probability for an adversary node to successfully guess the IDs of the set of nodes which form a circuit to be uniformly distributed in the network, that is:

- given a set `N = {n, n_1, ..., n_i-1}` of `i` nodes that exist in the network in a given time;
- a set of nodes `R = {n, n_1, ..., n_l-1}` which form an onion circuit with a set of nodes of length `l`,
- the length of the circuit `l`;
- and `Pr(n)`, which represents the probability that an adversary can successfully guess that a node `r` is a relay chosen to be part of a circuit, must be:

  ```
  Pr(n) = l/sum(N)
  ```

The probability must hold even though:

1) the node `n` does cannot have a full picture of the netowork at a given time and;

2) the adversary node knows the overlay fingertable of `n` at any time and;
3) relay nodes enter and leave the network in unpredictable ways (churn).

Two approaches to against route capturing and bias attacks when getting information to construct the onion circuit are:

1) Identify and isolate nodes performing attacks; This is not trivial to achieve in a completely decentralized way (i.e. without relying on central certificate authorities);

2) Use mechanisms to find node IDs that do not rely on direct finger table sharing from other nodes. One naive example could be to passively listen to networks activity for a while and select the node IDs that are more likely to be honest. This approach has the problem of publicly narrowing down too much the subset of nodes that are eligible to be part of the circuit, since possible malicious nodes can learn which nodes are more likely to be picked by the circuit constructor when using this passive mechanism.

Onion circuit construction vulnerabilities

Selecting the nodes that will be part of the onion circuit in a completely decentralized manner is no easy tasks. (to explain: relays are not kept anywhere and should be queried as a peer joins the network or wants to build an onion circuit)

Open questions and future work

minimum required entropy:

secure and decentralized circuit construction:

misbehaving peers and incentives:

public key distribution:

sybil attacks and in-dht routing:

peer discovery and circuit building:

performance and overhead:

Previous work

Bibliography

(gpestana), Goncalo Pestana. n.d. "Metadata Resistant Dht - Github." *Metadata Resistant DHT - Github*. https://github.com/gpestana/notes/issues/8.

Syverson, Paul. 2013. "Why I'm Not an Entropist." In *Security Protocols Xvii*, edited by Bruce Christianson, James A. Malcolm, Vashek Matyáš, and Michael Roe, 213–30. Berlin, Heidelberg: Springer Berlin Heidelberg.