Securing Kademlia with a low overhead Public-Key Infrastructure

Orson Peters

February 17, 2013

Abstract

This paper outlines a low overhead protocol that creates a robust public-key infrastructure (PKI) to secure Kademlia. Presented here is a centralized solution that uses a certification authority that provides identity to nodes. If correctly implemented this solution implicitly resists common attacks aimed at Kademlia and can be extended in various ways to further limit malicious activity to a minimum. This solution is unprecedented in low overhead (both latency and bandwidth) and supporting a recursive/iterative routing hybrid.

1 Introduction

Kademlia [1] is a high-profile Distributed Hash Table (DHT). Used in many P2P applications (LimeWire, Gnutella, Overnet, EDonkey2000, eMule, BitTorrent, etc) it is the most used DHT around. For good reason, its nifty topology allows for fast, flexible and low-overhead routing. However, Kademlia does not concern itself with security, and is vulnerable to many attacks.

In this paper we expand on the work on Likir [2]. Similarly, we propose a centralized certificate authority (CA) trusted by all participating nodes. However, we reduce overhead by introducing optimizations such as smaller signatures, timestamps instead of nonces and a novel way to pick randomized node ids. Through the use of secure request tokens and node ids we also enable a hybrid between recursive and iterative routing for lower latency while maintaining security.

A large portion of our optimizations come from the usage of Ed25519 [3]. It is a novel public key signing algorithm, with the desirable property of having small signatures and public keys (respectively 64 and 32 bytes). Furthermore, it is fast, highly secure and allows randomization of a public key without knowing the private key.

First we discuss which families of attacks Kademlia is vulnerable to. Then we propose the protocol, and how this differs from previous solutions. Afterwards we discuss how well the attacks on Kademlia are mitigated and which holes are potentially still left open. Last, we evaluate the overhead of the system and further improvements.

2 Terminology

Throughout this paper we use the following terminology:

$a||b$ - the concatenation of a and b
$H(x)$ - the SHA-1 hash of x
K^+ - the public key of the Ed25519 key pair K
K^- - the private key of the Ed25519 key pair K
$Sign(x, K)$ - the Ed25519 signature of x signed with the key pair K
$Verify(s, x, K^+)$ - verification of the Ed25519 signature s of message x with K^+

3 Attacks

Kademlia suffers from a wide range of attacks. In this paper we only discuss and prevent attacks against the network infrastructure - not network content. We do however provide a provable identity for every node, allowing various solutions like blacklists and reputation systems to secure content.

Man In The Middle. All unsecured internet protocols suffer from this attack. Because the transport layer is not secure any intermediate hop can change or drop any packet, and any party might be able to forge packets.

Replay attack. This is a more subtle version of the Man In The Middle (MITM) attack. An adversary eavesdrops some secured communication and stores it. Now if the communication does not contain any temporary resource such as a timestamp or nonce (number used once), the communication gets accepted when sent by the adversary - breaking the security. Worse, if the communication does not address a receiver for the content the adversary may *replay* the communication to any party. Replay attacks are attacks in its own right, but may be used to set up other attacks. For example, if a secured request for content is sent out, but does not contain the intended receiver for the request an adversary might send this request to many content providers, effectively creating a cheap and anonymous DDoS attack on the requester.

Routing attack. Each Kademlia node keeps track of a number of nodes to communicate with - the routing table. A routing attack is aimed at this table, filling it with either invalid, offline or even malicious nodes, hampering or even stopping the connectivity of the attacked node.

Eclipse attack. The Eclipse attack [4] is a routing attack where malicious nodes exploit the topology of the network to "surround" a victim node. If successful all or a majority of the routing of the victim node goes through the malicious nodes, allowing them to manipulate or block the communication.

Distributed Denial of Service attack. The Distributed Denial of Service attack (DDoS) is an attack where many computers flood one computer with requests, requiring so much resources the entire service is stopped or slowed down to a crawl.

Sybil attack. The Sybil attack [5] works by having one adversary provide many identities to the network, effectively gaining power. While it is not an attack on its own it can be used to greatly increase other attacks in effectiveness.

Kademlia suffers from all attacks described above. Our protocol is designed to explicitly or implicitly resist all of them.

4 The protocol

There exists a centralized certificate authority, the CA, that is known to and trusted by all nodes participating in the network. Furthermore, everyone knows the CAs public key, CAK^+.

Because we use timestamps instead of nonces is most locations in this protocol the network has a synchronized time, called the network time. While its not necessary to synchronize the time through the CA, it is the most practical and easily secured way of doing so - therefore we specify it in the protocol.

4.1 Synchronizing time

Before anything, a node should synchronize it's time to the network time. A node wanting to get the network time must generate a nonce, and send it to the CA along with some synchronize time opcode. The CA then replies with $nonce, network\_time, sign(nonce||network\_time, CAK^+)$. The node then verifies the nonce and the signature. Furthermore, the node must keep track of the time the request took (the round-trip-time). If this is more than a certain timeout, say 5 seconds, the node must reject the reply. Last, if everything was valid a node should substract half the round-trip-time from the time as a mean of crude accuracy improvement.

Because the reply is signed and contains a nonce all active MITM attacks and replay attacks are futile. Because the node rejects old replies a MITM can not delay a response in an attempt to desynchronize the network time. Assuming the worst case where a MITM maximized the delay the network time is desynchronized by at most $timeout/2$ seconds, which is very acceptable.

4.2 Getting a certificate

After synchronizing to the network time a node needs a certificate from the CA in order to be able to participate in the network. The CA signs the nodes identity, as well as a public key of the node, allowing the node to subsequently prove ownership of the certificate to peers.

First the node must generate a new public/private keypair.

References

[1] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," *Peer-to-Peer Systems*, pp. 53–65, 2002.

[2] L. Aiello, M. Milanesio, G. Ruffo, and R. Schifanella, "Tempering kademlia with a robust identity based system," in *Peer-to-Peer Computing, 2008. P2P'08. Eighth International Conference on*, pp. 30–39, IEEE, 2008.

[3] D. Bernstein, N. Duif, T. Lange, P. Schwabe, and B. Yang, "High-speed high-security signatures," *Cryptographic Hardware and Embedded Systems–CHES 2011*, pp. 124–142, 2011.

[4] A. Singh *et al.*, "Eclipse attacks on overlay networks: Threats and defenses," in *In IEEE INFOCOM*, Citeseer, 2006.

[5] J. Douceur, "The sybil attack," *Peer-to-peer Systems*, pp. 251–260, 2002.