S/Kademlia:
A Practicable Approach Towards Secure Key-Based Routing

Ingmar Baumgart, Sebastian Mies

P2P-NVE 2007, Hsinchu, Taiwan

Universität Karlsruhe (TH), Institute of Telematics
Prof. Dr. M. Zitterbart
Motivation

• Kademlia is a widely used Peer-to-Peer protocol
  • Azureus and eMule/eDonkey has over 1 million users
  • Proven to be scalable in “reality”

• But: No security considerations
  • Vulnerable to attacks → corrupted files, lookups fail
  • What about applications besides file-sharing?

• Our contribution:
  • Enhance Kademlia with security features
Kademlia Protocol

- Symmetric and unidirectional XOR metric
  - Lookups to converge to the same path
  - Allows reactive routing table maintenance

- Converging iterative parallel lookup
  - Iterative lookup $\rightarrow$ faster learning about new nodes
  - Asking nodes in parallel $\rightarrow$ Detection of failed nodes

- Simple: Only two RPCs needed for KBR
  - FIND_NODE and PING
Taxonomy of Attacks

- Underlay network
  - Spoofing, Eavesdropping, Packet modifications
- Overlay routing
  - Eclipse attack
  - Sybil attack
  - Adversarial routing
- Other attacks
  - Denial-of-Service
  - Data Storage
Underlay Network Attacks

- No protection against
  - Spoofing, eavesdropping, modifications
  - Overlay must provide end-to-end security
- Simple solution: Use NodeID := H( IP + Port )
  - No authentication, problems with NAT
  - IP spoofing still possible
- Better solution: Cryptographic NodeID
  - NodeID := H( public-key )
  - Allows authentication, key exchange, signing messages
Eclipse Attack

- Attacker: Cuts off a part of the network
  ➔ Lookups fail, data corruption, partitioning

- Countermeasure:
  Prevent a node from choosing its ID freely
Sybilattack

- Attacker: Adds huge number of nodes
  - Network under control

→ Countermeasure:
  Prevent joining with a huge quantity of nodes
S/Kademlia: Secure NodeID Assignment

- **NodeID** := H( public-key )
  - Allows secure communication between two nodes
  - Duplicate NodeIDs improbable

- **Signatures**
  - **Weak**: Signature( timestamp, IP, port )
    - Used for PING or FIND_NODE messages
  - **Strong**: Signature( message )
    - Used for DHT storage messages

- **Certificates**
  - Certificate of a well-known trusted CA
    - CA prevents sybil- and eclipse attack
  - Decentralized with crypto-puzzles
S/Kademlia: NodeID Crypto-Puzzles

Generate key pair
\( s_{publ}, s_{priv} \)

Calculate
\( P := H(H(s_{publ})) \)

NodeID := \( H(s_{publ}) \)

Choose a random X

Calculate
\( P := H(\text{NodeID} \oplus X) \)

Impedes eclipse attack

Impedes sybil attack

\( \text{No} \)

Preceeding \( e_2 \) bits in \( P \) zero?
Cryptographic NodeIDs for E2E security

Crypto-puzzles
- make it difficult to generate
  - a large quantity of NodeIDs (Sybil attack)
  - non-random NodeIDs (Eclipse attack)
  - Adversarial nodes uniformly distributed
- adapt when computational resources become cheaper
Adversarial Overlay Routing

- Attacker: Reroutes packets into own subnet
  → Lookup finds the closest adversarial node

  ![Diagram showing adversarial nodes and countermeasure]

  - Countermeasure: Use parallel lookups over disjoint paths
Let \((h_x)\) be the CDF of path lengths \(m\) the density of adversarial nodes and \(d\) the number of disjoint paths then a lookup succeeds with probability

\[
P_K := \sum_{i=1}^{\lfloor (h_x) \rfloor} \left( h_i \cdot \left( 1 - (1 - (1 - m)^i)^d \right) \right)
\]
Iterative lookup
- allows to ensure lookup on disjoint paths
- needs enhancement

Lookup procedure (simplified)
1. Lookup k closest nodes in own routing table
2. Distribute them over d path lookups
3. Do parallel path lookups
   - Check if a node already visited/result used
S/Kademlia: Lookup over disjoint paths

Source

Adversarial Node

Destination
• Motivation
  • Lookup probabilities analyzed for node lookups only
  • DHT operations need siblings for replication
    → A reliable sibling lookup is important

• Each node needs to know s closest nodes (siblings) to an identifier if it falls inside its sibling range
  → Not part of the original Kademlia design
A special sorted sibling list is introduced in S/Kademlia.

If $s$ is the number of siblings then a sibling list of size $>5s$ is needed at least to ensure that the node knows at least $s$ siblings to an identifier in its siblings range w.h.p (see proof in the paper).

→ Special splitting of smallest sub-tree can be omitted.
• Network join: lookup own NodeID
• Bucket Refresh
  • Bucket has not been used for a long time
    ➔ Lookup a random NodeID from the bucket
• Filling Routingtable with NodeIDs
  • Add nodes, if they are actively known
    (by a response to a RPC)
  • Add nodes, if they are passively known
    (by a RPC request) only if the prefix of X bits does not
    match the nodes NodeID

➔ Makes it difficult for an attacker to actively modify
routing-table
Evaluation: Simulator & Parameters

- Simulations have been done with OverSim
  - Provides a framework for overlays in OMNet++
  - Common KBRs already implemented
  - Open for contribution
  - Released under GPL
  - Visit [http://www.oversim.org](http://www.oversim.org)

- Simulation Parameters
  - SimpleUnderlay was used
  - No churn was simulated
Evaluation: Assumptions & Procedure

- **Assumptions**
  - Lookup fails if adversarial node is on lookup path
  - Lookup origin and destination not adversarial

- **Simulation Procedure**
  1.) Creation of a stable Overlay of with N Nodes
  2.) Lookup of N random nodes
  3.) Increase of adversarial nodes by 5%, repeat lookups
  4.) 90% of the nodes adversarial? → Stop
Evaluation: Results

Even with 25% adversarial nodes 99% lookups succeed.
Conclusions and Future Work

• Our design makes it difficult for an attacker to
  • launch large-scale sybil- or eclipse attacks and
  • enhances lookup success significantly in the presence of adversarial nodes
→ Not limited to Kademlia

• S/Kademlia has a sibling list
  • Lookup probability same as it is for the node lookup
  • Allows DHTs to reliably store replicas

• Future Work:
  • Evaluate security features under churn
  • Enhance S/Kademlia with a secure DHT
Thank you for listening! ... Questions?

This research was supported by

German Federal Ministry of Education and Research as part of the “ScaleNet” Project and the BW-FIT support program by the Landesstiftung Baden-Württemberg as part of the “SpoVNet” Project
Backup Slides
Evaluation: $N=10000, k=2d, d=\{1,2,4,8\}$

Fraction of successful node lookups

Fraction of adversarial nodes ($N=10000, s=16$)

Path length ($N=10000, s=16$)
Evaluation: \( N=10000, k=16, d=\{1,2,4,8\} \)

- Fraction of successful node lookups
- Fraction of adversarial nodes (\( N=10000, k=16, s=16 \))
- Path length (\( N=10000, k=16, s=16 \))
• Consists of \( n \) buckets of size \( k \). The \( i \)-th bucket holds nodes of distance \( 2^i \leq d(X, a) < 2^{i+1} \)
  
  → Each bucket covers a part of the identifier space

• Filling the routing table
  
  • First, only one bucket covering the whole identifier space exists
  
  • When a message from node \( X \) arrives and the Bucket which covers the identifier of \( X \) ...
    
    ▶ is not full: Add node \( X \) to bucket
    
    ▶ is full: Split the bucket, if it covers the own NodeID and add the node

• Special treatment of “close” nodes is needed
Introduction:
A Kademlia Routing Table Example

- Routing table with bucket size of k=2

![Routing table diagram with bucket size of k=2](image)