Secure P2P Routing
Overlays; e.g. Pastry

Elements:

- Nodes
- NodeIds
- Objects
- Keys
- Routing table
- Neigbor set
- Replica keys
Pastry; Nodes

Every **node** has;

- a unique 128-bit NodeId
- a routing table
- a neighbor set
- is a root for a key
Every **Object** has;

- a unique 128-bit key (same space as the nodeId)
- a root
- replicas (for fault tolerance)
Pastry; Routing table

- Keys and Nodelds are a sequence in base \(2^b\)

- Routing table contains \(128/2^b\) rows and \(2^b\) columns.

- The \(2^b\) entries in row \(r\) of the routing table contain the IP addresses of nodes whose nodelds share the first \(r\) digits with the present node’s nodeld

- A routing table entry is left empty if no node with the appropriate nodeld prefix is known.
Pastry; Routing

- Every step;
- Forward message to a node which shares at least one digit more with the key in the prefix than the present node.

- Or forward message to a node which shares the same prefix as the present node, but is numerically closer.

- If no node is found, the present node or the neighbor is the destination.
Attackers view

1.) Influence nodeId assignment.

2.) Influence routing tables.

3.) Influence message flow.
Nodeld Assignment

Randomly assignment = security

Hashing the IP address.
Ca's (Certified Authority)
Crypto puzzle.
Ca's

- Ca is not involved in regular operations. (e.g routing).
- Only nodes with valid certificates can join the overlay.
- Multiple nodeId certificates per IP address.
- Works good for Chord, Tapestry or Pastry but not CAN
New nodes have to generate a key pair with the property that the SHA-1 hash of the public key has the first $p$ bits zero.

The expected number of operations required to generate such a key pair is $2^p$.

$x = \text{SHA-1(\text{SHA-1(ipaddr,x)},\text{nodeId})}$ has $p'$ bits equal to zero. Nodes would be required to present such an $x$ for the pair $(\text{nodeId},\text{ipaddr})$ to be accepted by others.

$P$ is a value for security.

--> very hard for an attacker to gain a lot of NodeIds.
Influence Routing Table

Bad routing table/neighbour set updates are like a bad NodeId assignment.

Overlay using network proximity can attacked by this. (Pastry, Tapestry)

After every update, the probability that a faulty node is in the routing table, should be $f$, but it is $(1-f)f + f*1$
constrained routing table

Two Routing tables;
- One for efficient routing (uses network proximity)
- One for secure routing (constraint that a nodeId should be closest to some point in the id space)

- Secure bootstrap nodes for new nodes.

- When normal routing fails, uses secure routing.

- To have a faulty node in the routing table is at $f$
- $(not (1-f)*f +f*1)$
message flow

- Ca`s and a second routing table prevent attacks on the routing algorithm and the nodeId assignment.

- If faulty nodes doesn't forward messages, a overlay with a fraction of $f$ faulty nodes is massively disturbed.

$$(1-f)^{h-1}$$
detect faults, use diverse routes

The idea;
Defining a failure test.
If failure test is positive use the secure route over the constraint table

If failure test is negative everything is ok.
Failure test

• Each correct node $p$ computes the average numerical distance, $\mu_p$, between consecutive nodeIds in its neighbor set.

• The neighbor set of $p$ contains $l + 1$ live nodes: the $l/2$ nodes with the closest nodeIds less than $p$’s, and the $l/2$ nodes with the closest nodeIds greater than $p$’s

• To test a prospective root neighbor set, $rn = id0, .... , idl+1$, for a key $x$, $p$ have to check 2 things.
Failure test

1. all nodeIds in $rn$ (Route neighbor set) have a valid nodeId certificate, the closest nodeId to the key is the middle one, and the nodeIds satisfy the definition of a neighbor set.

2. the average numerical distance, $\mu_{rn}$, between consecutive nodeIds in $rn$ satisfies: $\mu_{rn} < \mu_p \times \gamma$

The parameter $\gamma$ controls the tradeoff between $\alpha$ and $\beta$.

$\alpha$ .... false positive.
$\beta$ .... false negative
The alternative route

1.) $p$ sends $r$ messages to the destination key $x$. Each message is forwarded via a different member of $p$'s neighbor set; this causes the messages to use diverse routes.

2.) Any correct node that receives one of the messages and has $x$'s root in its neighbor set returns its nodeId certificate and the nonce, signed with its private key, to $p$.

3.) $p$ collects in a set $N$ the $\lfloor l/2 \rfloor + 1$ nodeId certificates numerically closest to $x$ on the left, and the $\lceil l/2 \rceil + 1$ closest to $x$ on the right. Only certificates with valid signed nonces are added to $N$ and they are first marked pending.

4.) After a timeout or after all $r$ replies are received, $p$ sends a list with the nodeIds in $N$ to each node marked pending in $N$ and marks the nodes done.

5.) Any correct node that receives this list forwards $p$'s original message to the nodes in its neighbor set that are not in the list, or it sends a confirmation to $p$ if there are no such nodes. This may cause steps 2 to 4 to be repeated.

6.) Once $p$ has received a confirmation from each of the nodes in $N$, or step 4 was executed three times, it computes the set of replica roots for $x$ from $N$. 
The alternative route
Any Questions?