Sybil Attack and Defense in P2P Networks

Presented by: Pratch Piyawongwisal, Pengye Xia

CS 538 Fall 2011

Advanced Computer Networks

Outline

- Sybil attack
- Attacks on DHTs
- Solutions
  - Using social networks (Whānau)
  - Limiting attacks in structured overlays (ACS)

The Sybil Attack

- A malicious user pretends to be multiple nodes in the system by faking identities
- Douceur [IPTPS 2002] was the first to consider this problem in the context of structured P2P networks

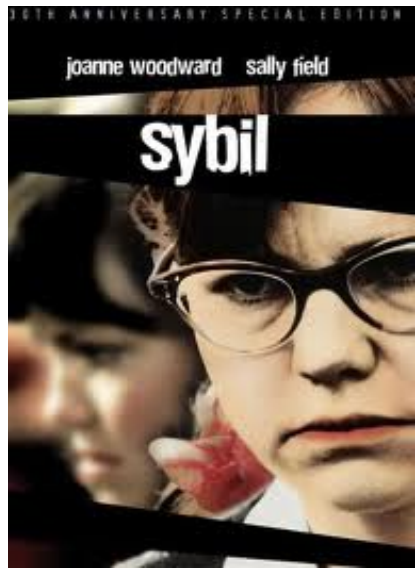


image source: http://www.filmaffinity.com/en/film769491.html

Targets

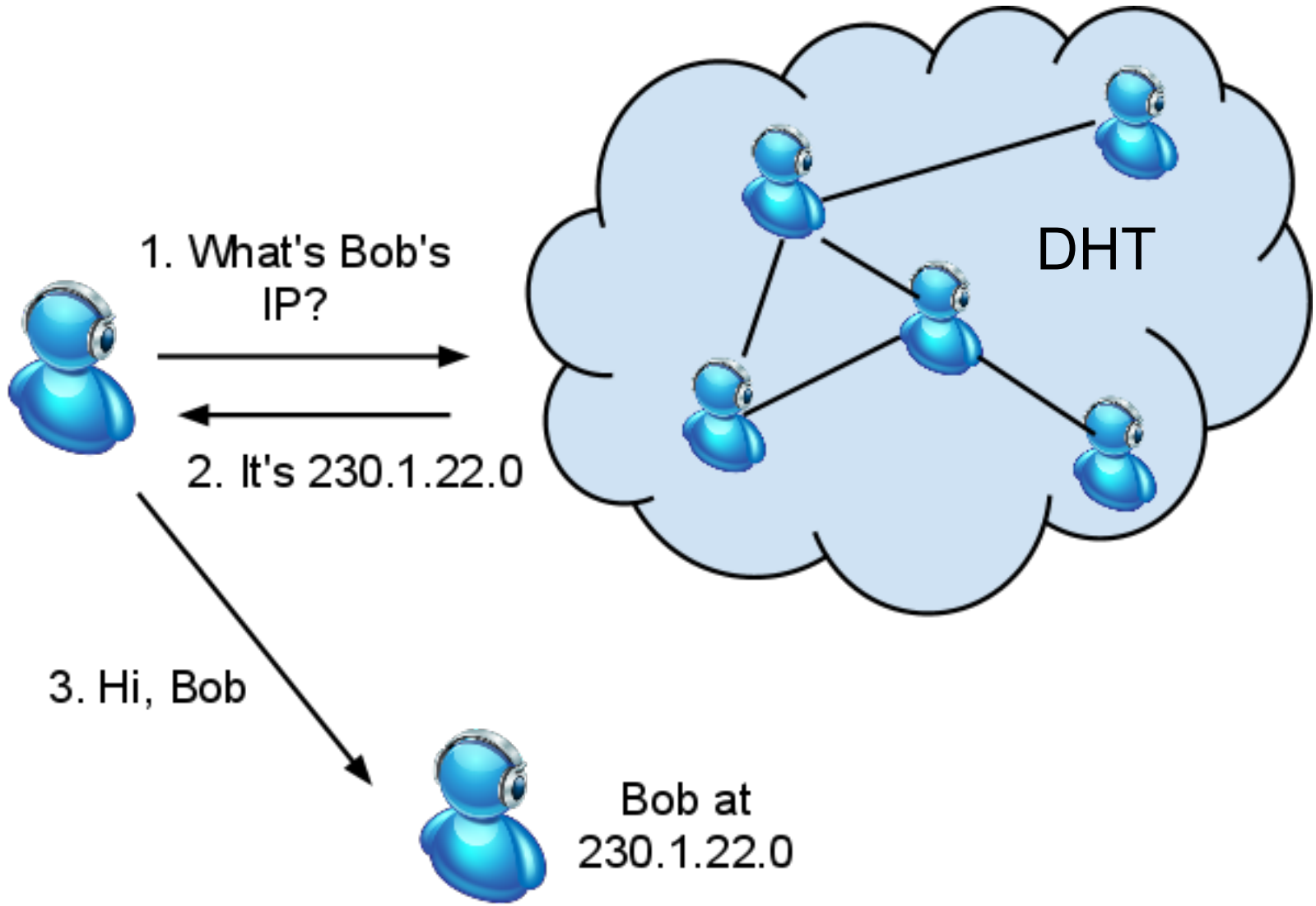
- Reputation system
  - eBay: create fake accounts to give positive feedback to a seller
- Internet polls
  - use multiple IP addresses to get more votes
- eMule
  - use multiple IDs to attack the KAD network
- Google Page-Rank
  - create multiple sites that link to a target site to increase its rank

Proposed Solutions

- Trusted Authority
  - Require all nodes to authenticate with public keys
  - Douceur, 2002: Without a centralized authority, Sybil attacks are always possible
  - Centralized, expensive
- Resource Testing
  - Tests computing power, storage, network bandwidth, limited IP addresses
  - e.g. CAPTCHA, cryptographic puzzles
  - Adversary may have more resources

Sybil attack on a Distributed Hash Table (DHT)

Scenario: Instant Message (IM) application

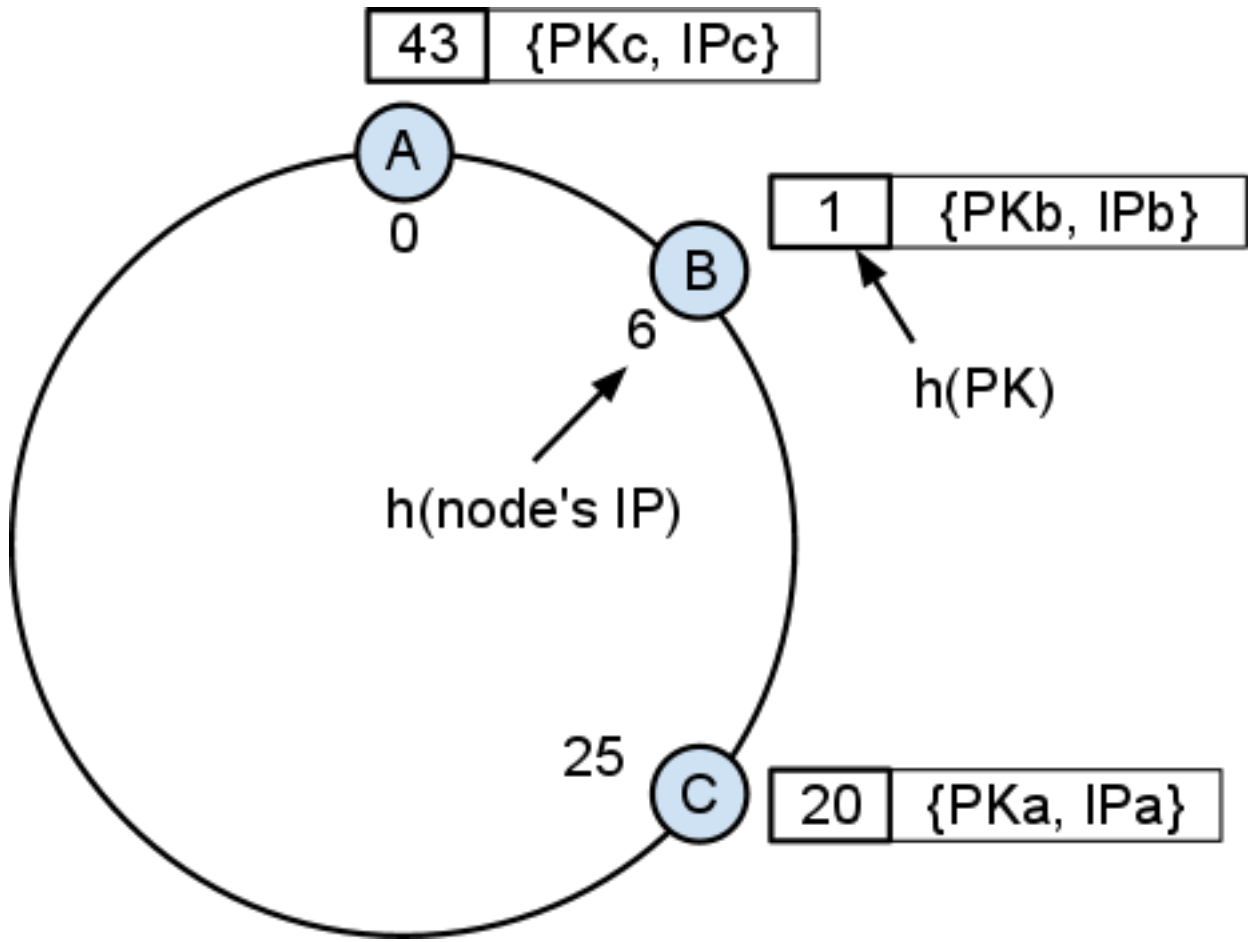


Scenario: Instant Messaging (IM)

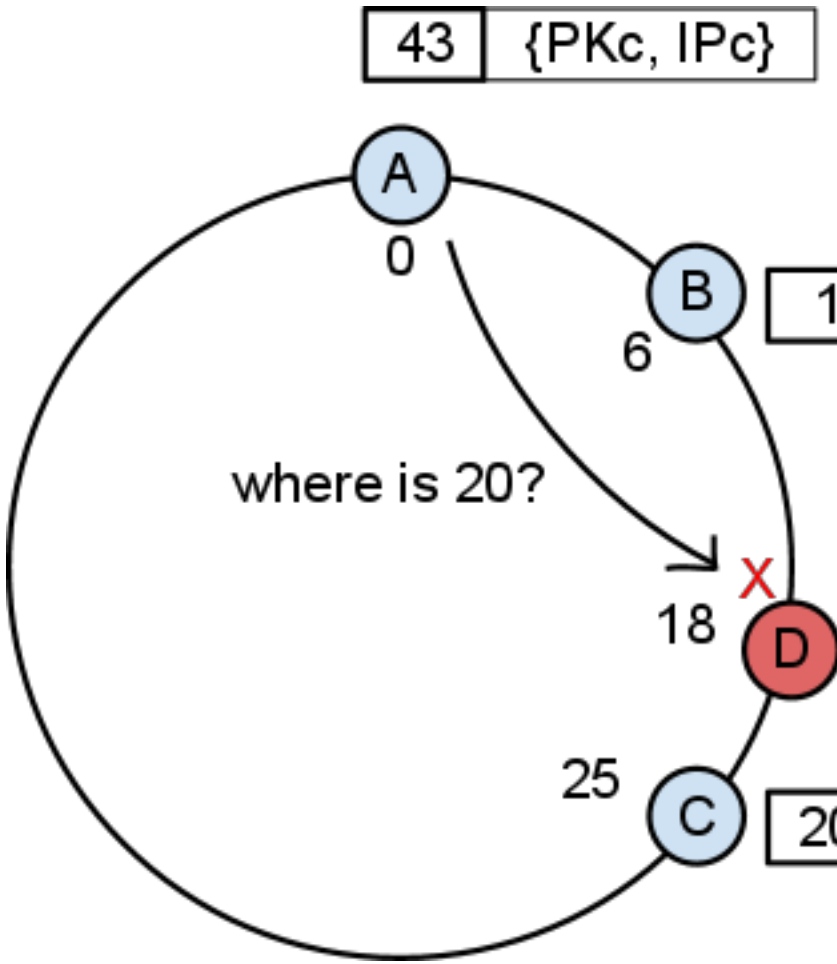
- At the beginning, each user has
  - KEY: a public key PK
  - VALUE: self-signed tuple {PK, IP address}
  - everyone else's PK
- Goals:
  - A wants to talk to B
  - Given PK_B, A can retrieve $\{PK_B, IP_B\}$ by querying any node in the DHT
- Concerns:
  - DHT-related metrics (e.g. lookup time, table size, etc.)
  - Protection against attacks

Scenario: IM application

- How the Chord handles this situation:



1. Attack on DHT routing



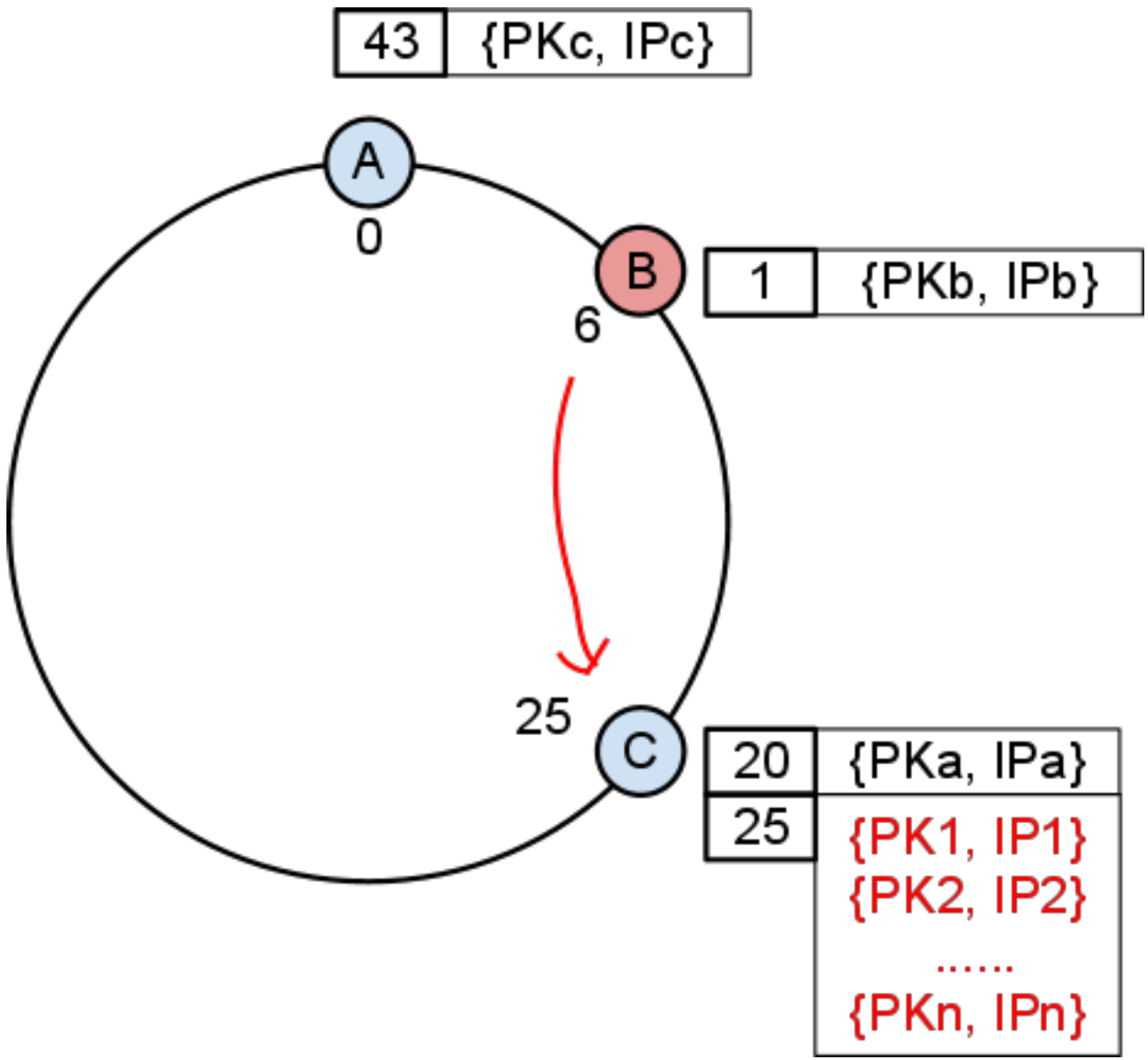
Attacker simply drops queries or return false information

2. Clustering attack on specific node

Attacker targets at node C

It inserts *n* objects into DHT, whose keys all hash to 25. These objects are stored at C.

Overflow at node C!

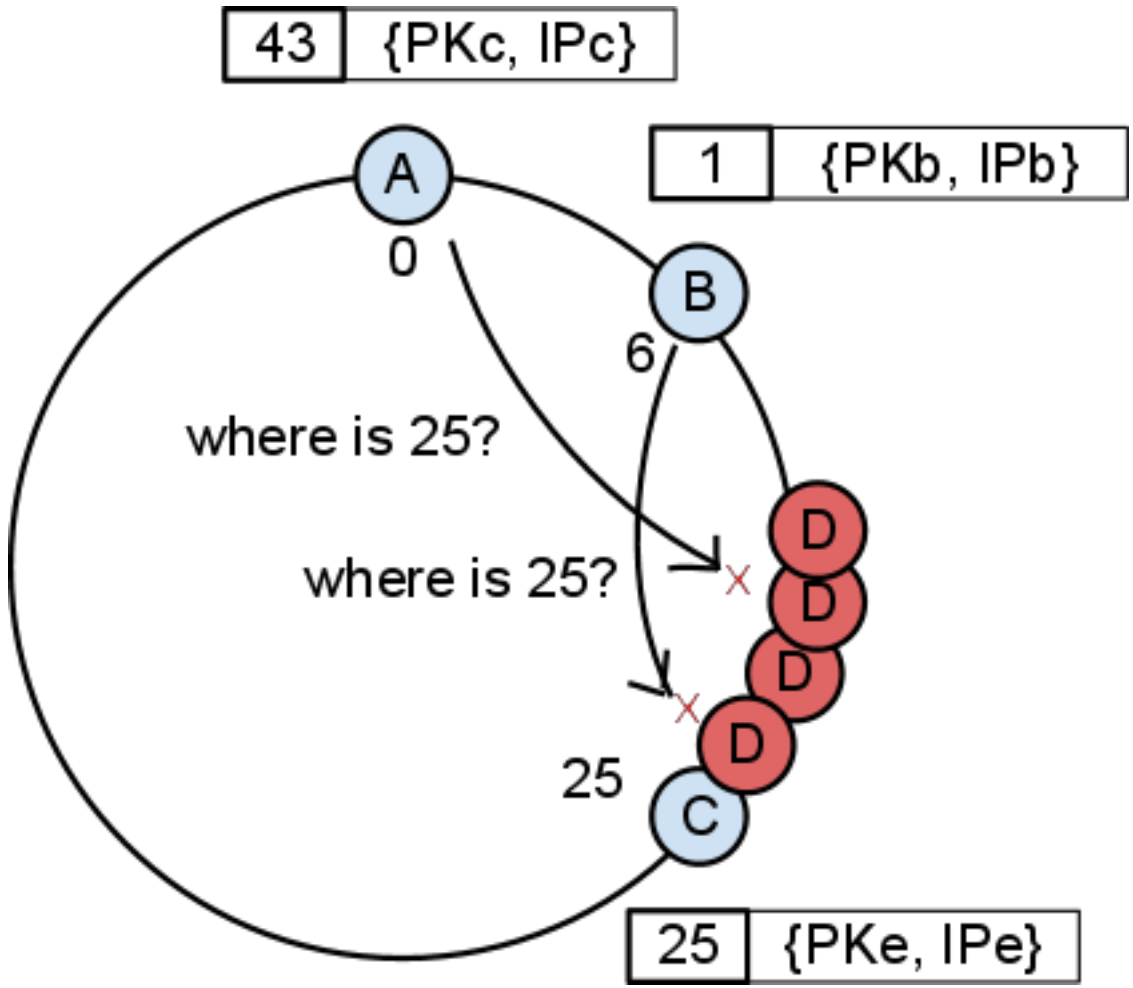


3. Clustering attack on specific key

An attacker wants to intercept the query on PK_E (key ID 25)

It creates many sybil nodes around key 25 and increases possibility of intercepting queries

Queries on PK_E will never be answered!



Summarize: Types of attacks on DHT

- Targeting at DHT routing
  - Attacker create many sybil node IDs
  - Response queries with nothing or false information
- Targeting at specific node
  - Guess the hash function h(key)->ID
  - Insert many keys with h(key) near the specific node ID
  - Overflow!
- Targeting at specific key
  - Guess the hash function h(key)->ID
  - Create many sybil node IDs near the key ID
  - Affect queries on the key

Whānau: A Sybil-proof Distributed Hash Table

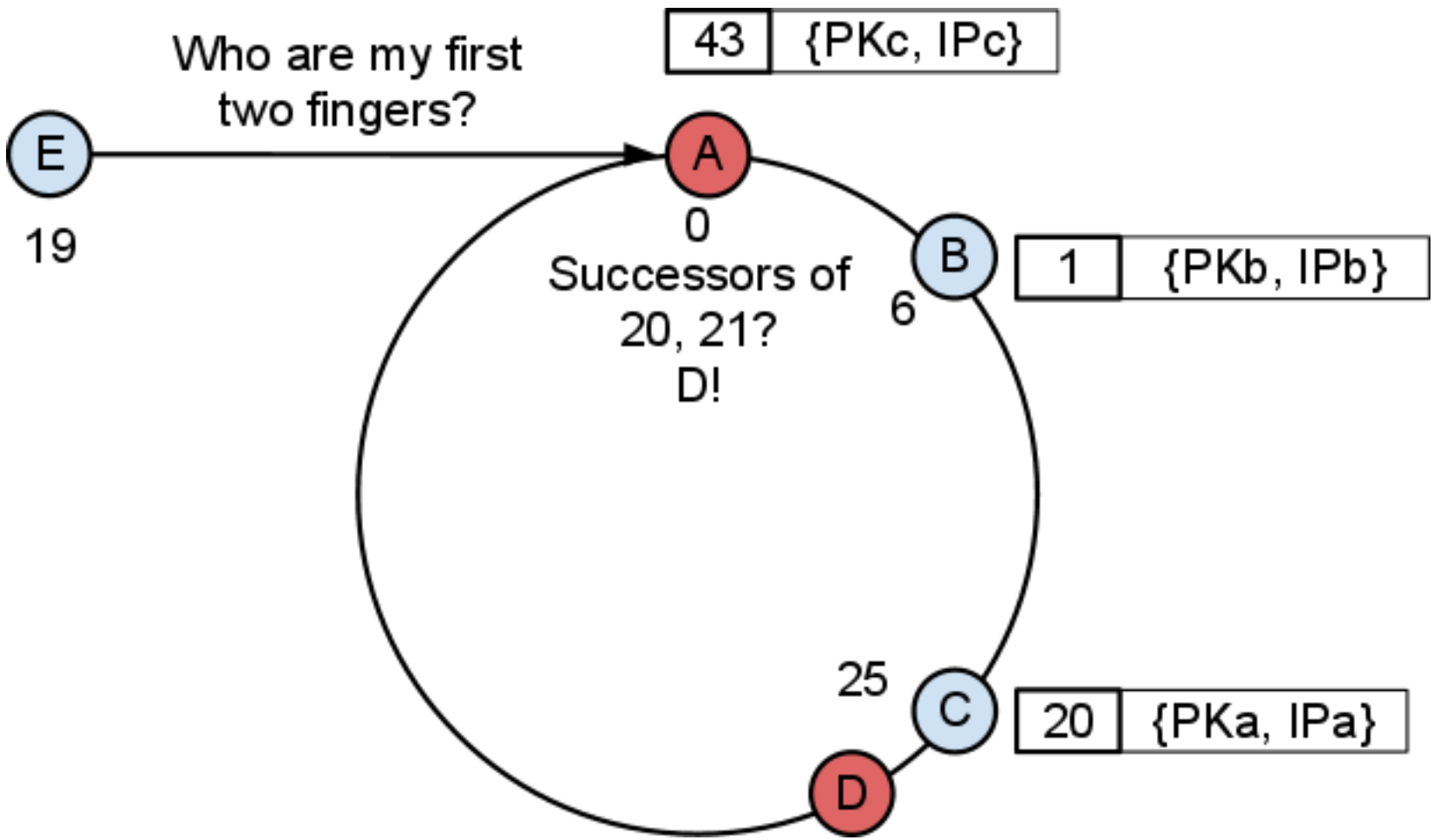
C. Lesniewski-Laas and M. F. Kaashoek, MIT CSAIL [NSDI 2010]

Vulnerability of Chord

- Vulnerable to Sybil attacks
- Influence of attackers increase with time!
  - Table maintenance is done through queries
    - Sybil nodes make responses with false info to fill sybil IDs into honest node's finger table
  - Influence of Sybil nodes increases during each node join, leave and failure
  - Example: node join

Example: node join

A new node E joins the DHT



Some Intuitions

- Don't build or maintain table by queries
  - Prevent influence of Sybil nodes from increasing
- Don't rely on a single finger
  - Reduce the impact of Sybil fingers

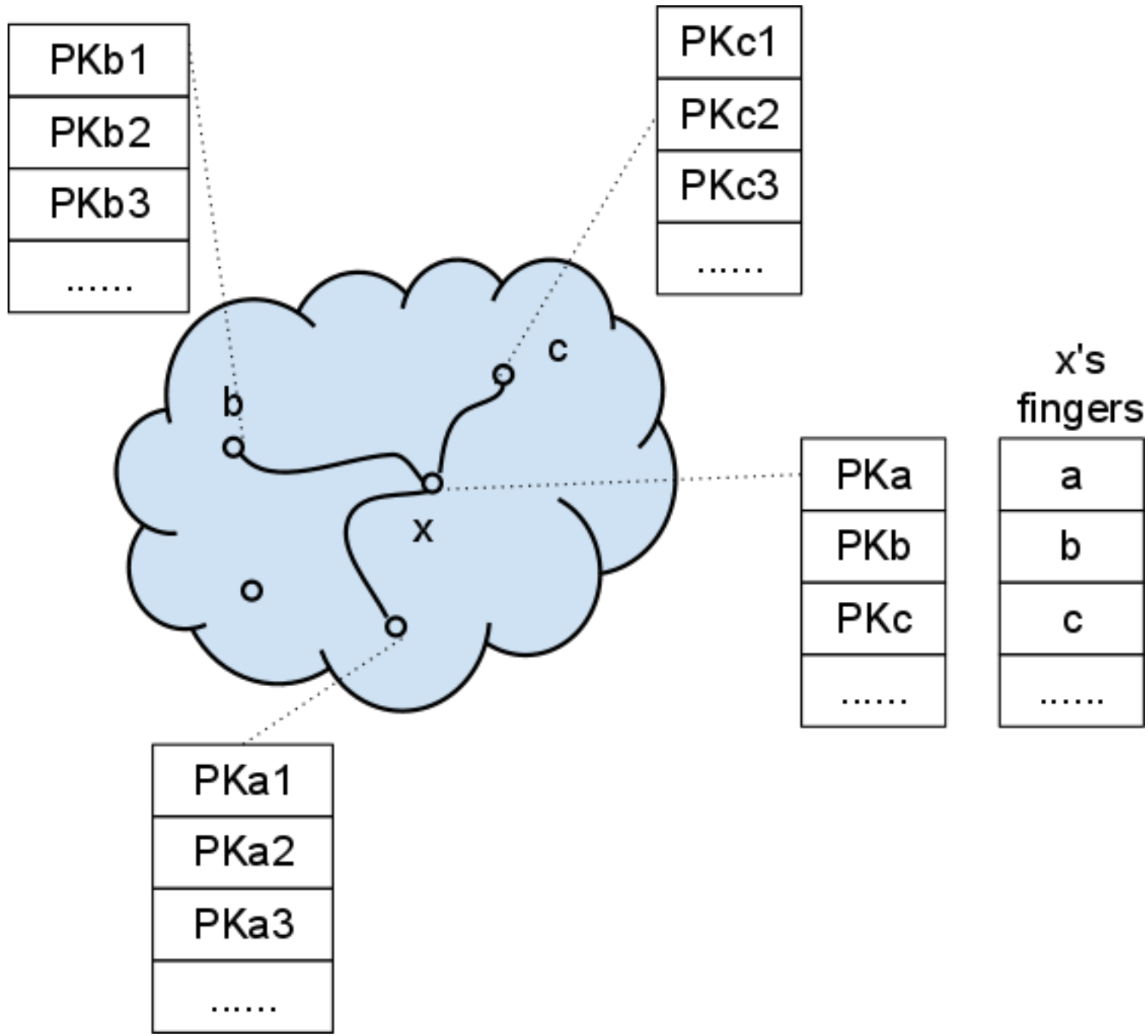
- Strawman protocol
  - Build table by random sampling
  - Base for Whānau

Strawman protocol

Denote the set of local keys stored at a node by *local(node)*

If the union of {local(x)}, {local(a)}, {local(b)}, {local(c)} covers the whole key space, a key can always be found at *x*!

Requirement 1: Sampling must be random, sample size $O(\sqrt{n})$

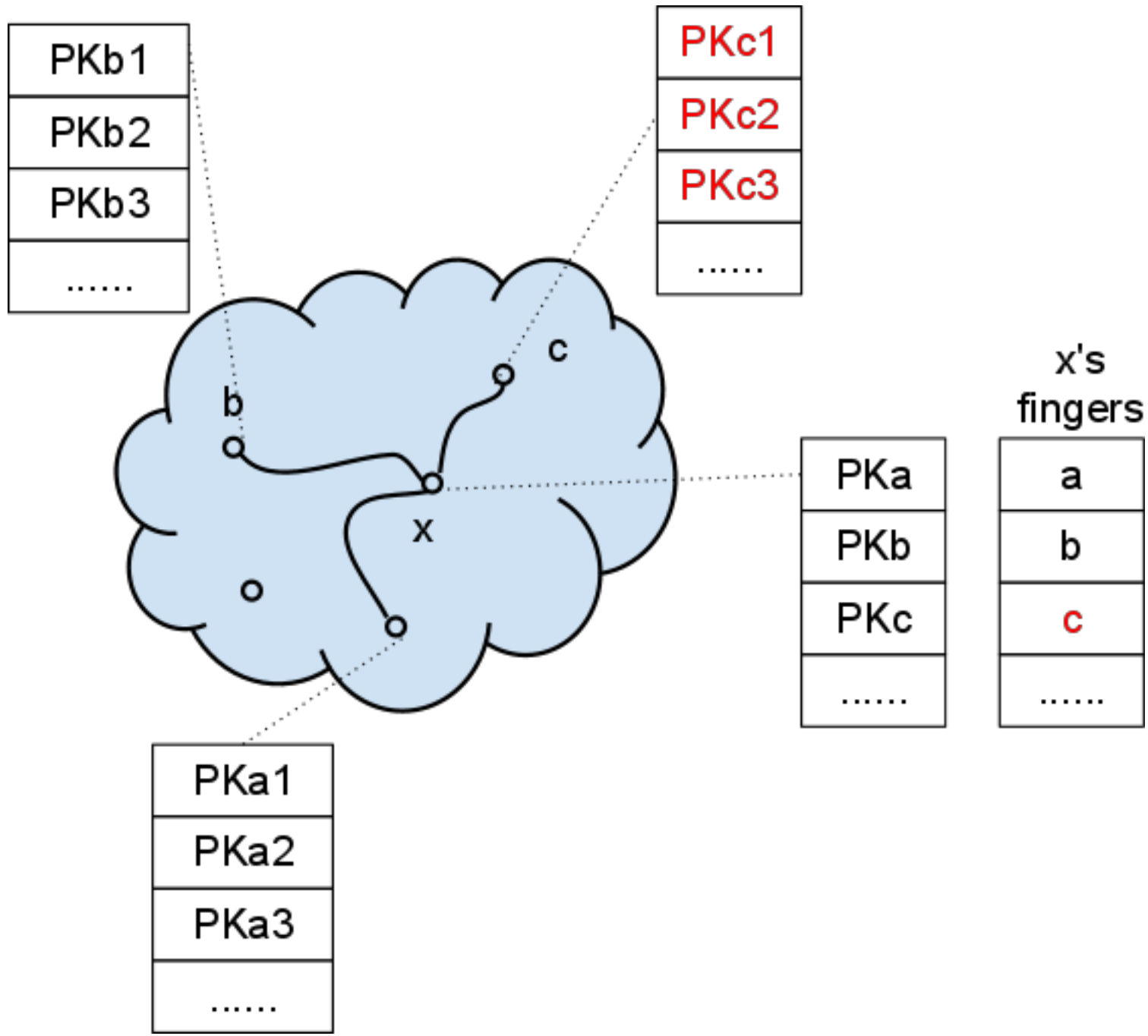


Strawman protocol

Denote the set of local keys stored at a node by *local(node)*

If the union of {local(x)}, {local(a)}, {local(b)}, {local(c)} covers the whole key space, a key can always be found at *x*!

Requirement 2: Limited percentage of sybil fingers in the finger table



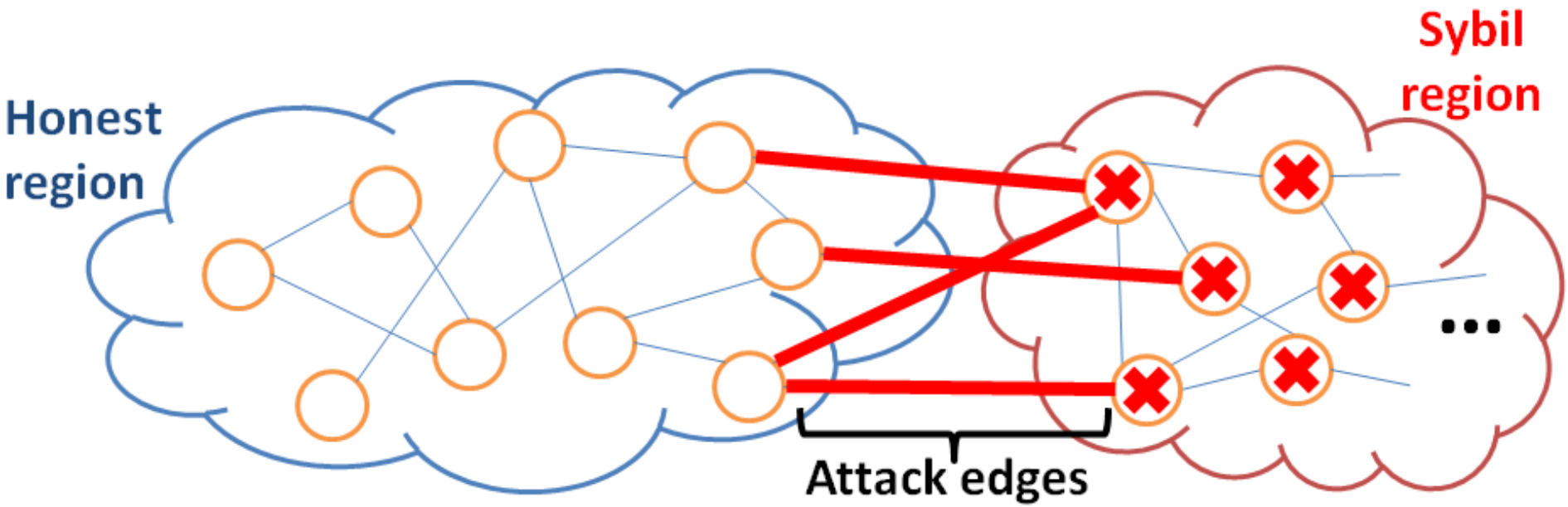
Strawman protocol

- As long as node can do random sampling to build its table and limit the number of Sybil fingers in the finger table, Strawman protocol is not vulnerable to Sybil attacks
  - Reason: local tables of honest fingers cover the whole key space

- How?
  - Sampling by random walks on a social network

Social Networks

- Node have social links to other nodes
  - provides additional information



Sampling by random walks on the social network

- Assumption:
  1. The number of attack edges is much smaller than the number honest edges
  2. Fast mixing
- Properties:
  - Due to 1, random walk has higher probability to land on a honest node
    - Percentage of Sybil nodes in a random sample is limited
  - Due to 2, given enough length of walk, the probability of ending at any node is uniformly distributed
    - Sampling is random

Random sampling in Strawman

- Start *t* random walks of length *w* on the social network
  - $t = O(\sqrt{n})$ is large enough for collecting enough honest fingers
  - $w = O(\log n)$ is large enough to ensure uniform distribution
  - *n* is the number of honest nodes
- Union of honest fingers' local table cover the whole key space

Strawman vs Chord

Scheme	Availability	Efficiency	Security
Chord	Always available in absence of attacks	O(log n) hops lookup; O(log n) message per query O(log n) table size	Vulnerable to Sybil attacks; Influence of Sybil nodes increases
Strawman	Always available	One-hop lookup; O(√ n) number of messages per query due to broadcast O(√ n) table size	Sybil attacks are mitigated; Influence of Sybil nodes does not increase

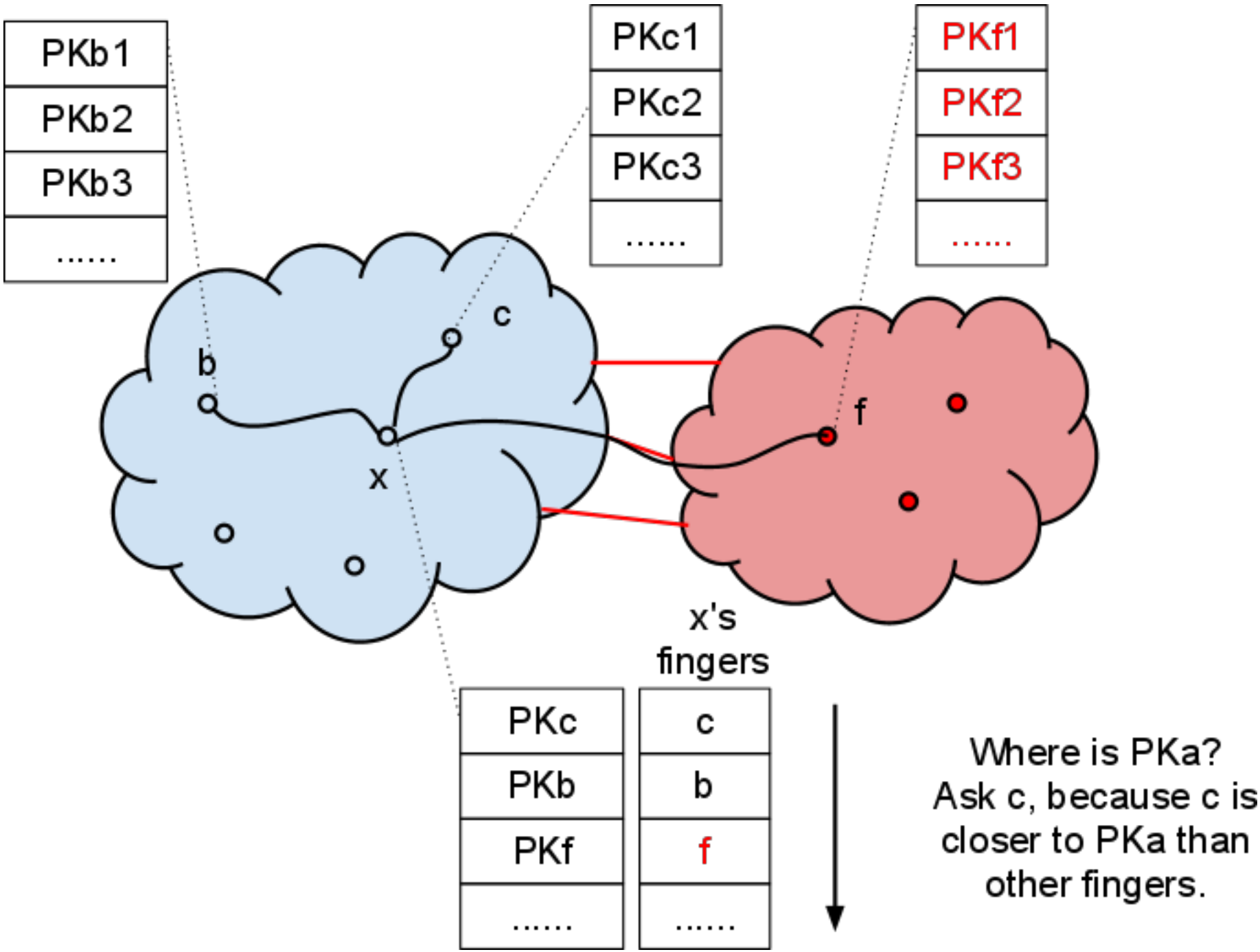
Combine the two?

Whānau*

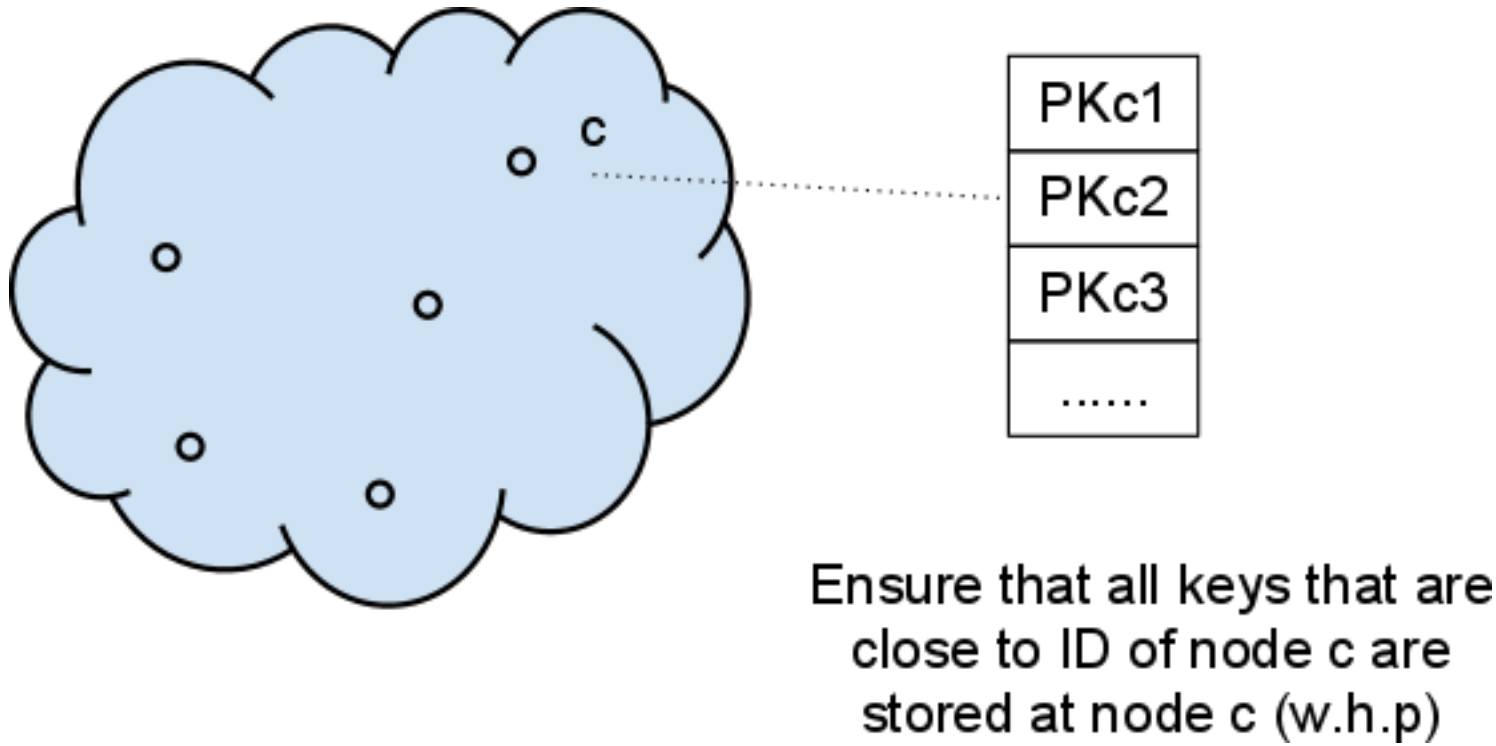
- Build and maintain routing tables using random walk on social network
  - Similar as Strawman
  - Influence of sybil nodes does not increase
- Ordering of fingers
  - fingers are ordered by the distance to the key, only a few finger are queried
  - O(1) messages per query

* Pronounced "far no"
  A Māori word meaning "extended family" or "kin"

How does Whānau work?



How does Whānau work?



Chord vs Strawman vs Whānau

Scheme	Availability	Efficiency	Security
Chord	Always available in absence of attacks	O(log n) hops lookup O(log n) message per query due to finger ordering O(log n) table size	Vulnerable to Sybil attacks; Influence of Sybil nodes increases
Strawman	Always available	One- hop lookup O(√ n) number of messages per query due to broadcast O(√ n) table size	Sybil attacks are mitigated; Influence of Sybil nodes does not increase
Whānau	Always available	One-hop lookup O(1) number of message on average O(√ n log n) table size	Sybil attacks are mitigated; Influence of Sybil nodes does not increase

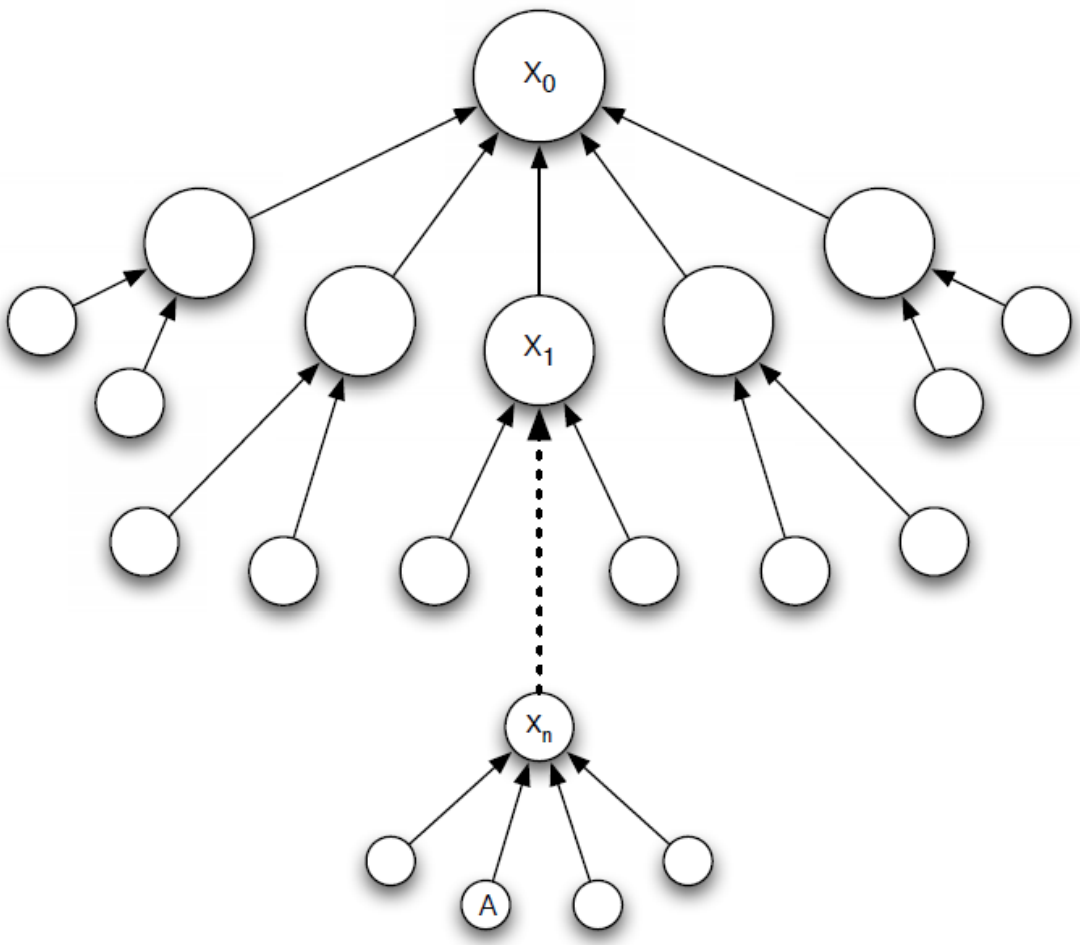
Limiting Sybil Attacks in Structure P2P Networks

Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta.
[INFOCOM 2007]

Limiting Sybil Attacks in Structured P2P Networks

- Main goal
  - Limit the rate at which a node can obtain IDs
  - Differ from Whānau that tries to limit the influence of Sybil nodes on DHT
- Admission Control System (ACS)
  - Maintain a tree hierarchy
  - Require the joining node to solve puzzles from leaf up to the root (Resource testing)
  - Once authenticated, the joining node becomes a leaf

ACS: Authentication



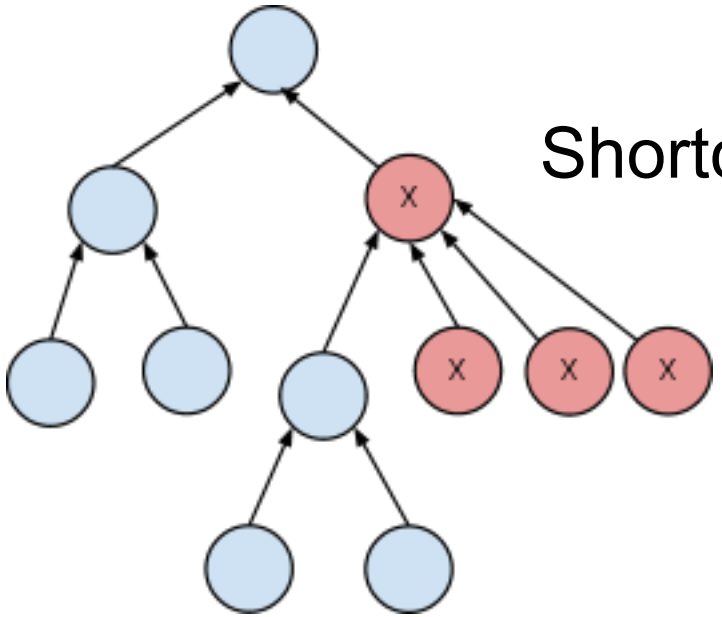
The puzzle:

Given PK, TS, H, solve for R s.t.
h(PK, TS, R) = H

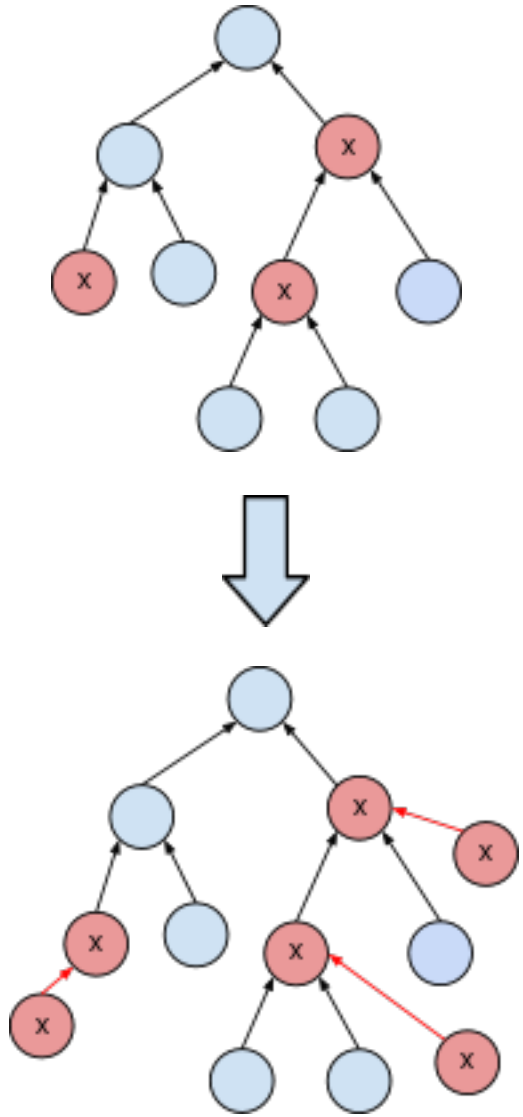
A wants to join:

1) A asks a leaf X_i for puzzle
2) A solves the puzzle and receives a token from X_i
3) A can now ask one upper level X_{i-1} for another puzzle
4) A keeps solving puzzles until it the root
5) Finally, A is assigned a random ID_A, certified by the root

Attack and Defense on ACS

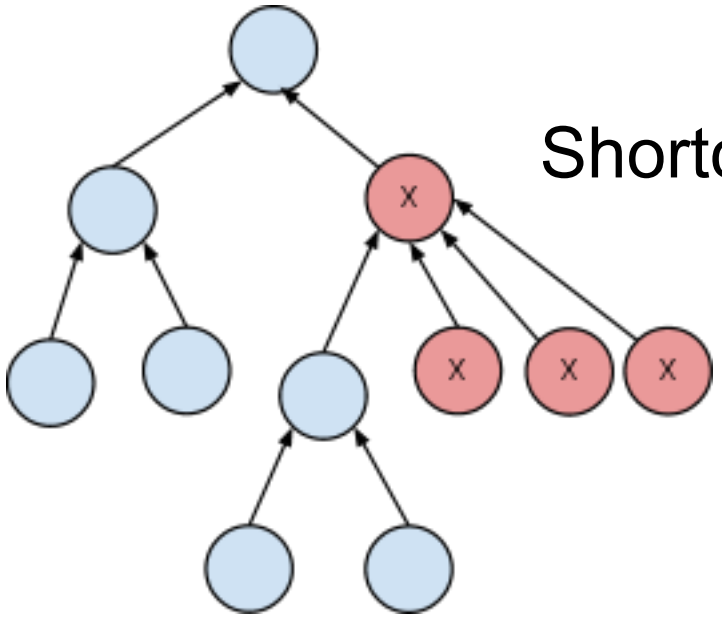


Shortcut



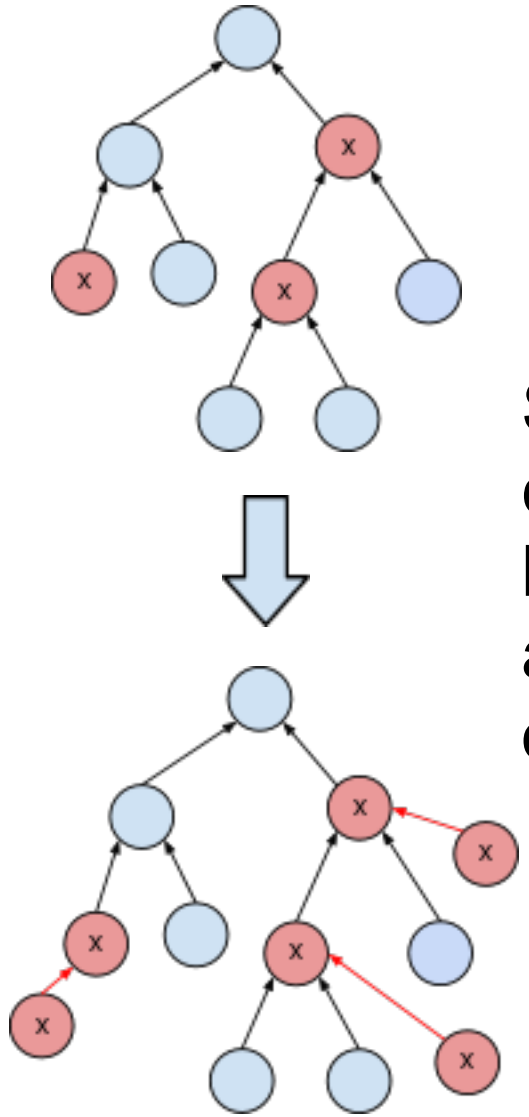
Sybil nodes can patiently but steadily accumulate over time

Attack and Defense on ACS



Shortcut

Solution:
Limit the token issuing rate



Sybil nodes can patiently but steadily accumulate over time

Solution:
Periodically redo the puzzle solving

ACS and Whānau: Different approaches in the fight against Sybil Attack

- Whānau: limit Sybil nodes' interference on DHT
  - Use its own customized one-hop DHT
  - Require social network and its special properties (i.e. sparse cut, fast mixing)
  - The number and strength of Sybil nodes does not matter.

- ACS: limit the number of Sybil nodes
  - Work with any DHT
  - Require trusted root (single point of failure)
  - Aim to minimize the number of Sybil nodes in the system
  - Cannot stop attackers with ample resource

Discussions

1. Whānau and Strawman protocol rely on the sparse attacking edges and fast mixing properties of the social network.
    - How realistic are these assumptions?
    - What kinds of social networks do they apply to?
    - How could we experimentally test these assumptions?

2. Is clustering attack on a particular node or key still possible with Whānau?
3. What is the advantage of having a tree structure as in ACS? Instead of going from the leaf to the root, can't we just pick a number of random nodes to authenticate with?

References

- C. Lesniewski-Laas and M. F. Kaashoek. **Whānau: A sybil-proof distributed hash table**. In Proc. NSDI'10, San Jose, CA, Apr 2010.

- Hosam Rowaihy, William Enck, Patrick McDaniel, and Thomas La Porta. **Limiting sybil attacks in structured p2p networks**. In 26th IEEE Intl. Conference on Computer Communications (INFOCOM), 2007.

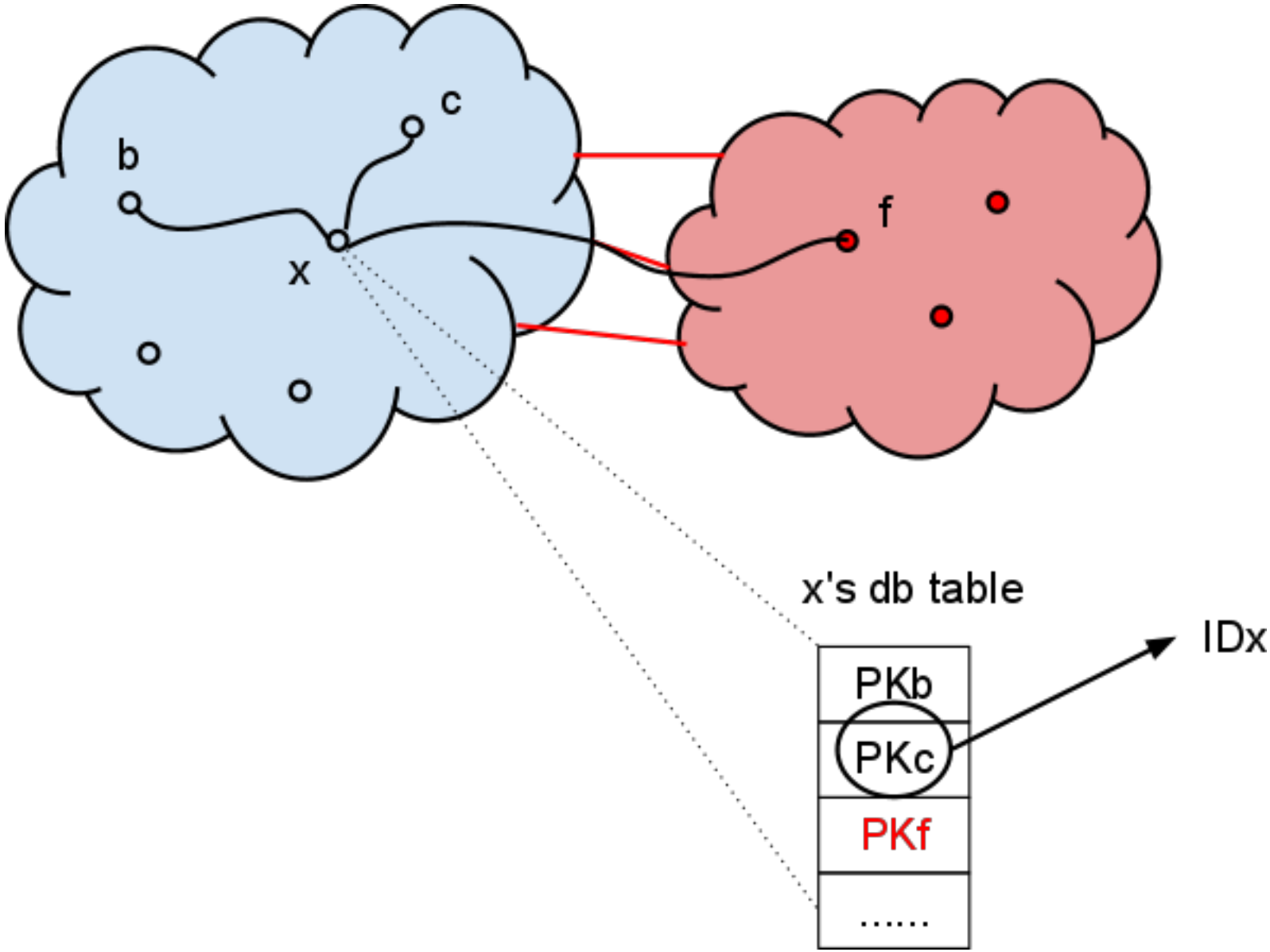
- B.N. Levine, et al. **A survey of solutions to the sybil attack**. Technical Report 2006-052, Univ. of Massachussets Amherst, 2006.

Appendix: Whānau in details

Setup: ID assignment

- Assign node ID:
  - Start r_d random walks, collect the key-value pair at the end of each walk and store the records locally.
    - To help build the successor table
  - Randomly select a key from the collected records as ID
    - To prevent node attack

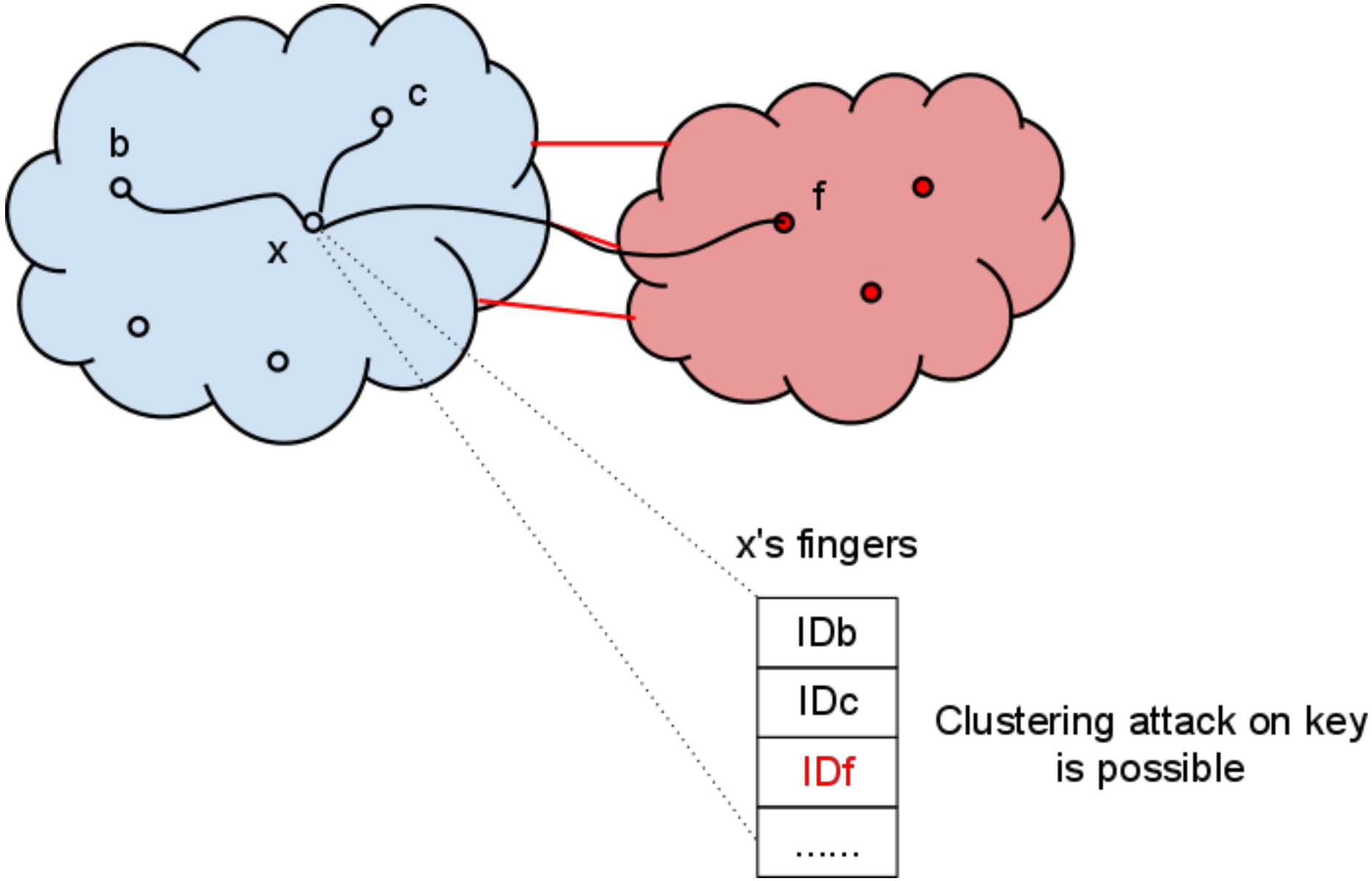
Assign ID



Setup: finger table

- Build finger table:
  - Start r_f random walks and collect node ID at the end of each walk
  - The collected node IDs form finger table

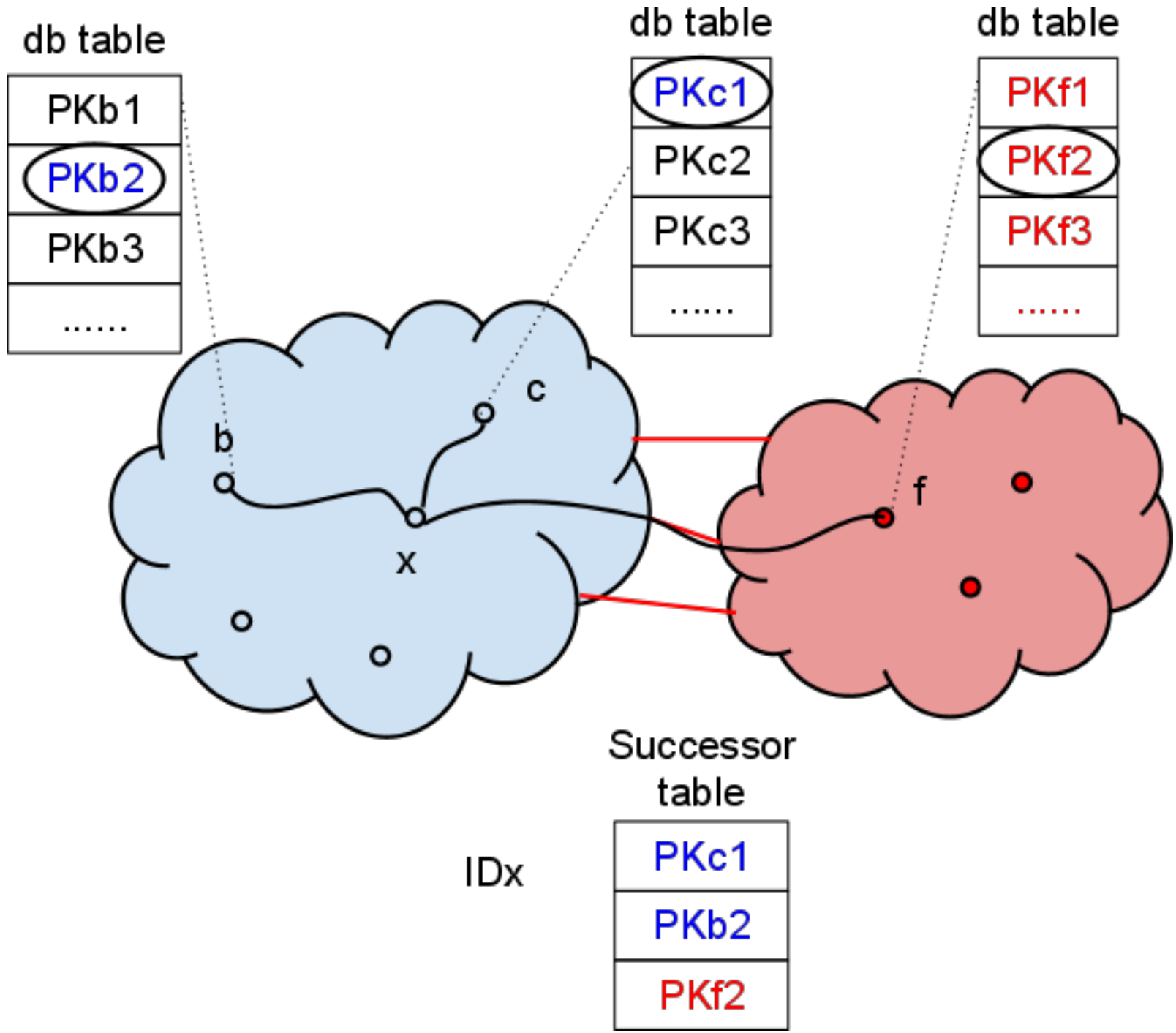
Finger Table



Setup: successor table

- Build successor table:
  - My Successor table contains all keys which are close to the my node ID
  - Start r_s random walks and collect a few key-value records closest to the my own ID from the end node's db table of each walk
  - The collected keys form successor table, the corresponding values are also stored locally
- Given enough number of random walks, all keys that are close to the node's ID has been stored into the successor table

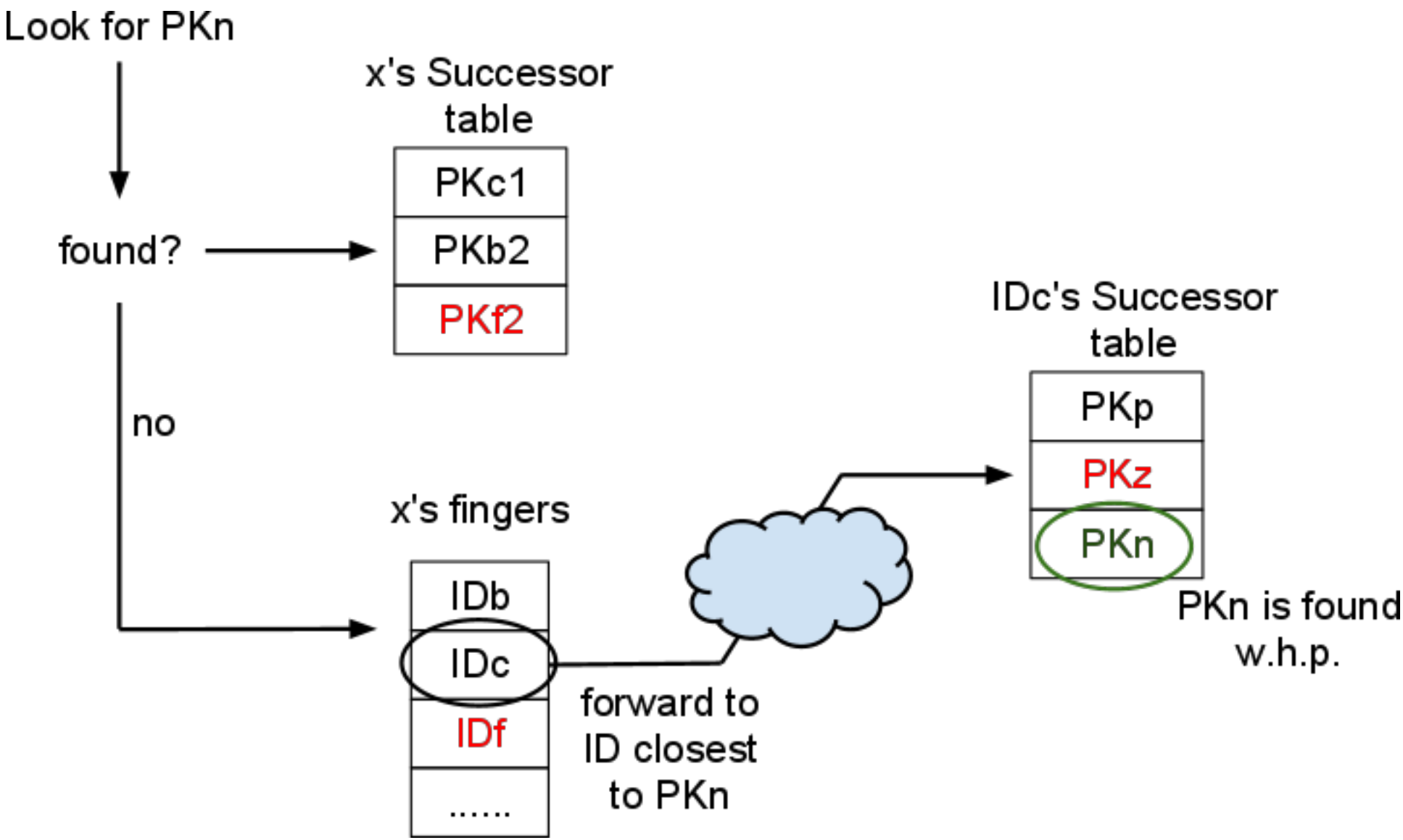
Successor table



Lookup

- Given a query on key *k*, a node *A*
  - First see if *k* is in its own successor table *succ(A)*
  - If not, find a finger *f* with closest distance to *k* in the finger table.
  - Ask if *k* is in *f*'s successor table *succ(f)*
- If *A* cannot find *k*, start a random walk from A and repeat the previous step on the end node of the walk
- Repeat the random walk process until we find the key

Lookup


Look for PKn
found?
x's Successor table
PKc1
PKb2
PKf2
no
x's fingers
IDb
IDc
IDf
.....
forward to ID closest to PKn
IDc's Successor table
PKp
PKz
PKn
PKn is found w.h.p.

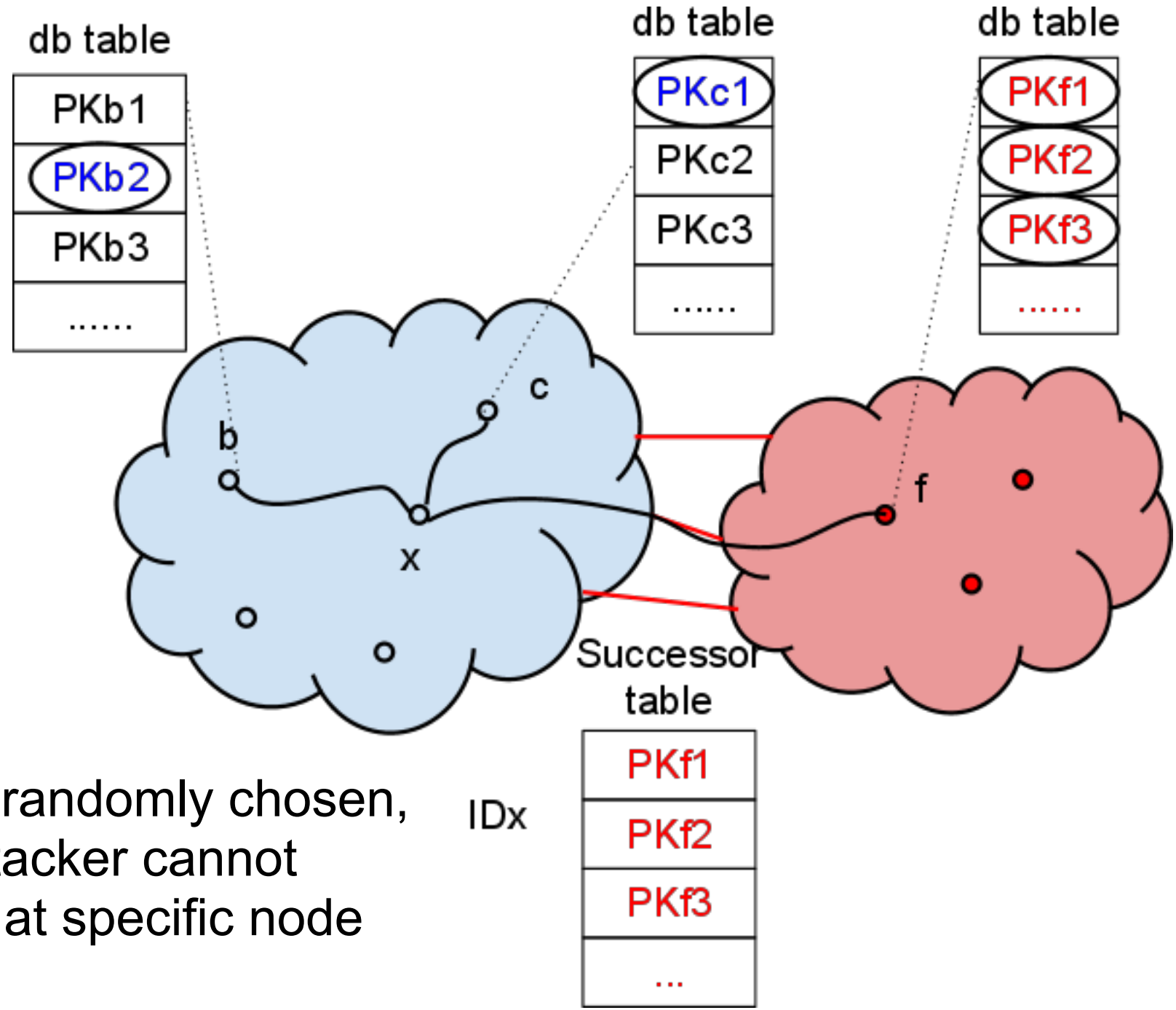
Availability and Efficiency?

- A key will always be found
  - Flooding argument
- One-hop look-up
  - By careful choice of r_d, r_f and r_s, the union of my successor table and all my honest fingers' successor tables cover the whole key space (w.h.p)
- On average O(1) messages and worst case O(log*n*) messages
  - Only ask the finger that is closest to the key instead of broadcasting

Security?

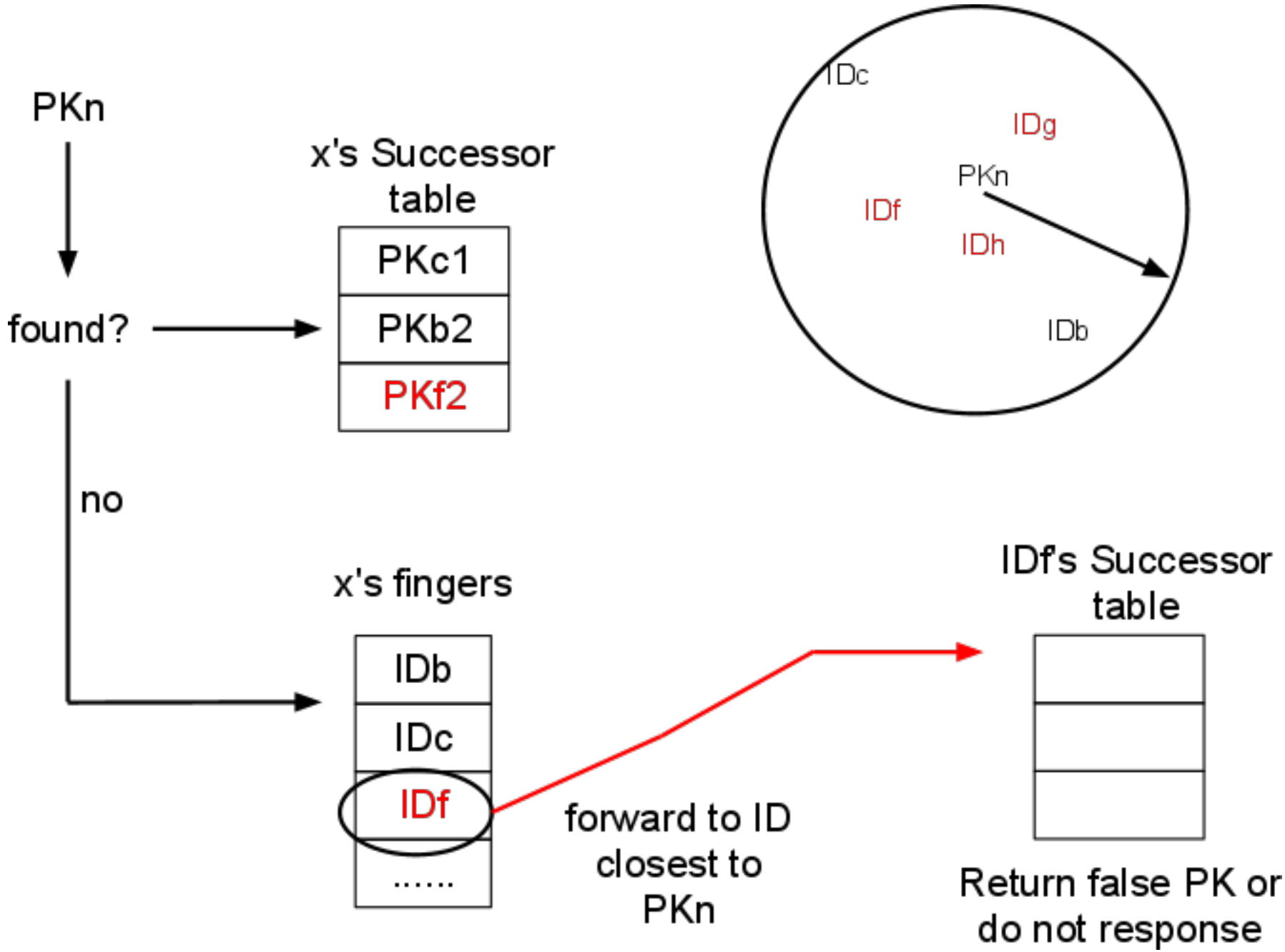
- Attacker's influence will not accumulate
- General attacks on DHT routing are mitigated
- Clustering attack on specific node is mitigated
  - If an attacker inserts many keys close to each other (using many sybil nodes), it is possible to overflow a certain node. But the attacker cannot decide which node he's attacking.
- Clustering attack on specific key is still possible
  - Example, an attacker creates many false IDs near specific key

Attack on node



IDx is randomly chosen, the attacker cannot target at specific node

Attack on the key

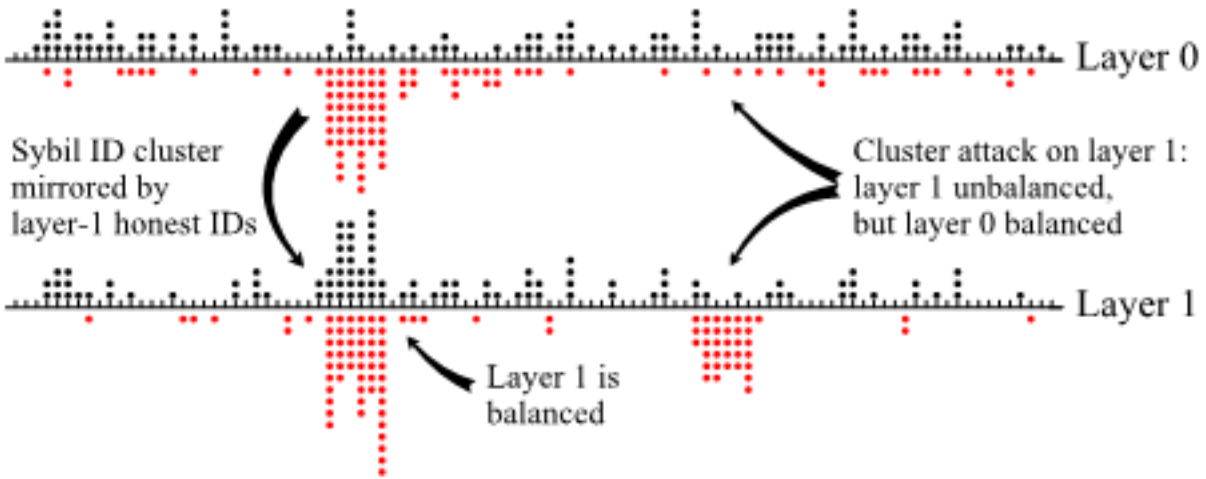


Defense against clustering attack

- The basic idea is to let distribution of honest IDs mimic the distribution of sybil IDs
  - Query will not be disturbed if around a certain key there are enough honest IDs
- Layered IDs:
  - Define the ID obtained by previous algorithm as layer-0 ID
  - Layer-1 ID is obtained by randomly picking a ID from the layer-0 finger table
  - Layer-1 finger table and layer-1 successor table is obtained similarly as before but using layer-1 IDs
  - Multiple layers are defined recursively

Distribution of node IDs

- If an attacker creates many nodes with layer-0 IDs near a certain key, it is highly possible that each honest node's finger table contains such an ID
- As a result, layer-1 IDs of honest nodes are highly possible to concentrate around the certain key



Modified lookup

- Given a query on key *k*, a node A
  - starts by looking at its layer-0 tables
  - if not found, randomly chooses a layer *i,* looks at layer-i table to find the key
- Suppose attacker's layer-j IDs concentrate around *k*, in all layers higher than layer-j, the honest node IDs will also concentrate around *k*
- Lookup in higher layers will not be affected by the clustering attack

How sparse should it be?

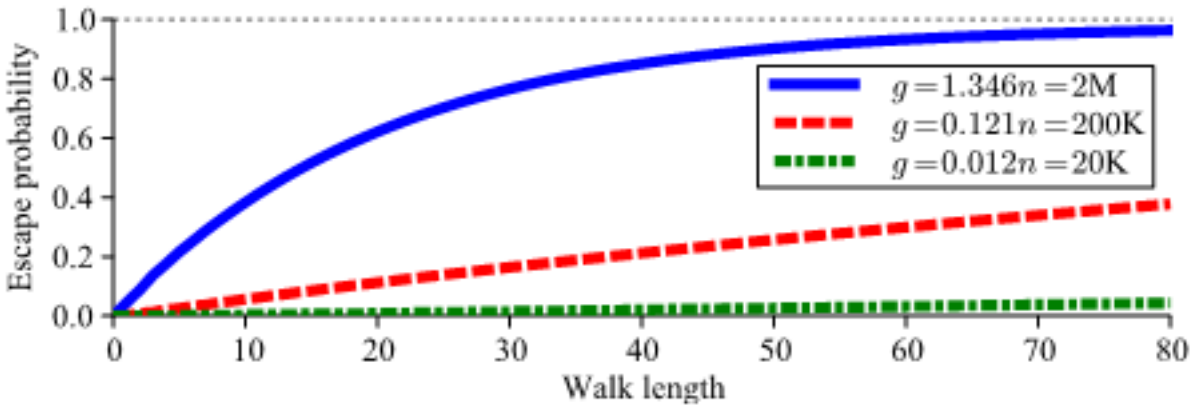


Figure 7: Escape probability on the Flickr network.

g: number of attack edges

n: number of honest edges

y-axis: probability of a random walk ending at a Sybil node

Example: node join

A new node E joins the DHT