Enhancing Network Privacy in Bitcoin

Aseem Baranwal, Ben Armstrong

Introduction

- Bitcoin is composed of nodes connected by a network
- Nodes generate transactions
- Transactions should be spread to all nodes in the network

Introduction

- Transactions in Bitcoin are linked to a specific account/node
- Accounts prefer not to be linked to their IP address
- Spreading protocols obfuscate the connection between IP and account
- Some protocols are vulnerable to attack

Outline

- Introduction
- Problem Statement
- Existing Spreading Protocols
- Our Contributions
  - Sync-Diffusion
  - Random-Diffusion
- Attack Model
- Evaluation
- Conclusions

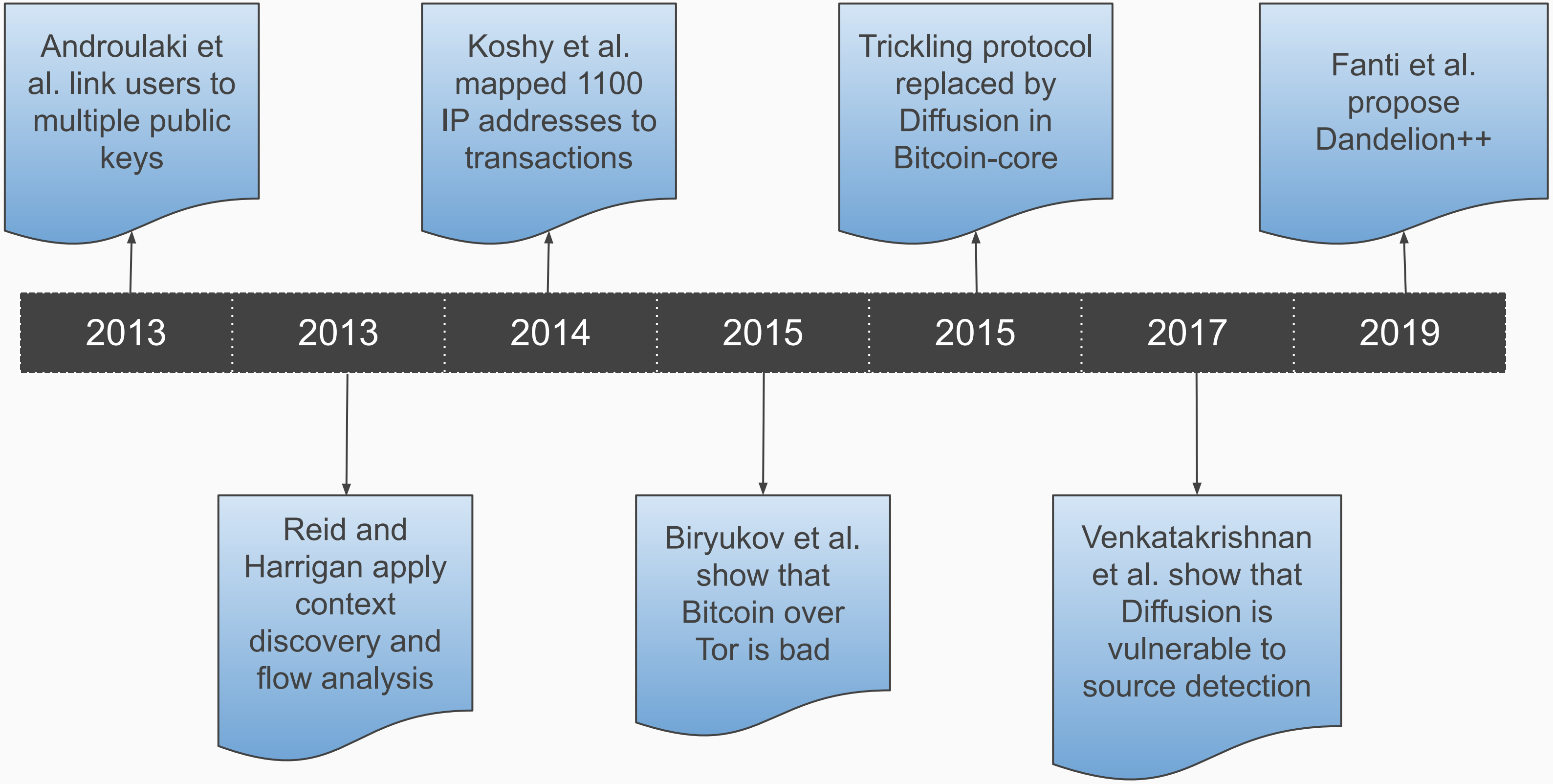
Problem Overview

- Bitcoin transaction spreading can be de-anonymized
- Cryptographic solutions only help on the blockchain layer (public keys are not real identities)
- Network layer solutions like Tor, I2P suffer from development issues and other problems (Monero, ZeroCoin)
- Goal: Design new performant protocols resistant to anonymity attacks

Why new protocols?

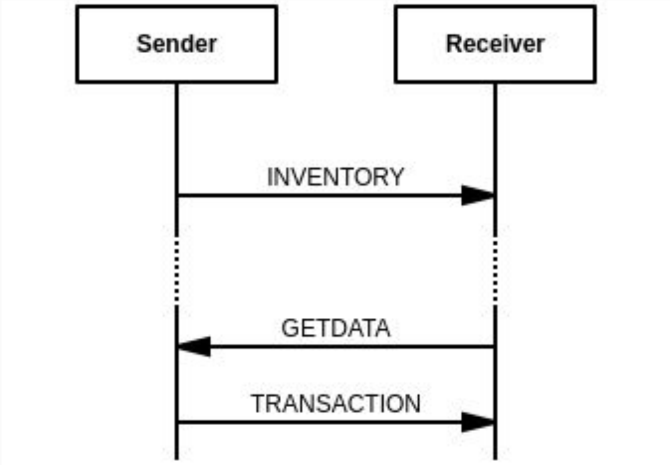
- Current protocols shown to be insecure (Trickling, Diffusion)
- Only two proposed alternatives
    - **Dandelion** (Venkatakrishnan et al., 2017) relies on unrealistic assumptions
    - **Dandelion++** (Fanti et al., 2019) has a vulnerable second phase

Related Work



Spreading Protocols

- Sender initiates INVENTORY message
- Receiver performs checks
- Receiver requests transaction via GETDATA message
- Sender sends TRANSACTION message



Trickling

- Used up to 2015
  - Rounds of 100 ms
  - Randomly chosen neighbor becomes “trickle node” in each round based on hashing
  - Transaction message propagated through this node
- Expected to randomly delay hops and hide the source of a transaction

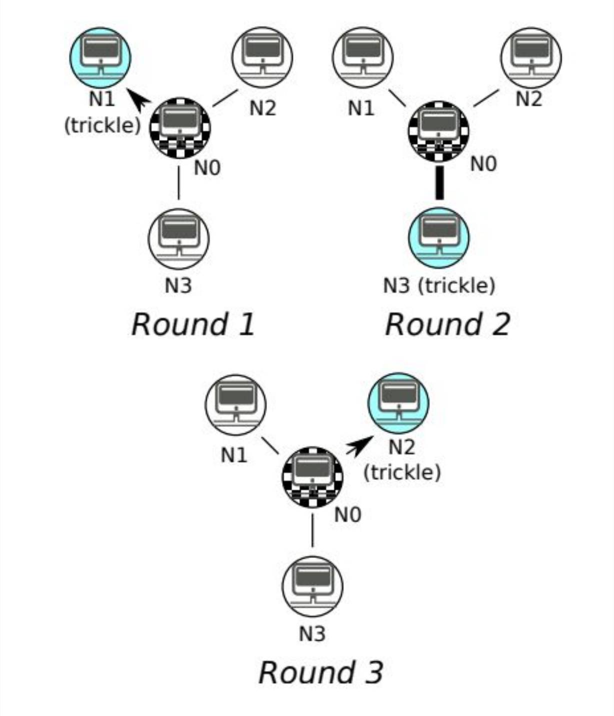
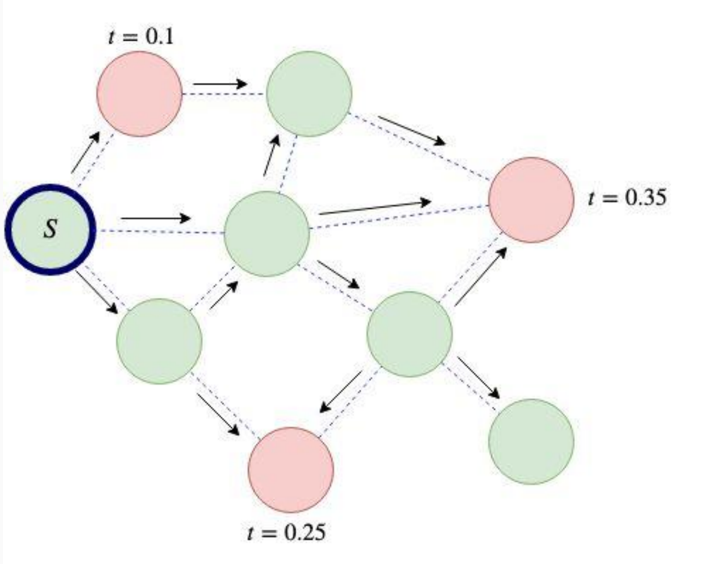


Illustration by Biryukov (2013)

Diffusion

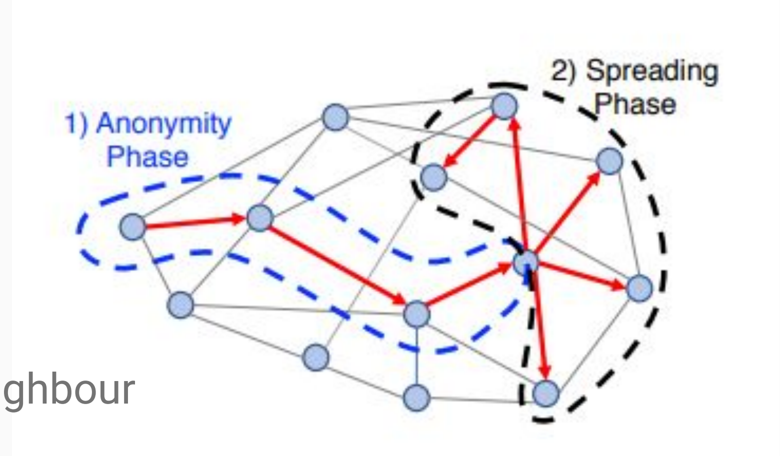
- New protocol by the community (2015)
- Based on ideas by Patrick Strateman
- Code changes in Bitcoin-core made by Pieter Wuille
- Random independent Poisson delays



GitHub PR: bitcoin/bitcoin/pull/7125

Dandelion

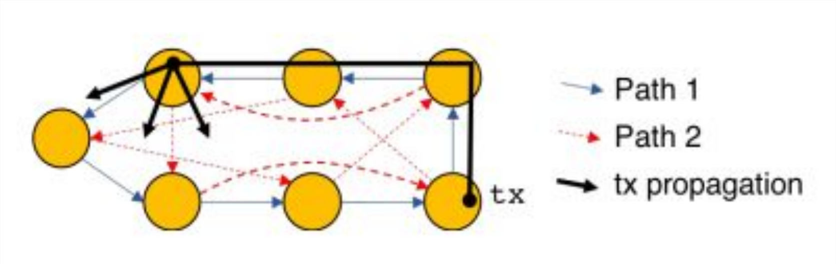
Alternative protocol proposal by Venkatakrishnan et al.



- Stem phase
  - Relay message to one random neighbour
  - Follows a line-graph
- Fluff phase
  - Start diffusion at a randomly chosen node
  - All subsequent nodes perform diffusion

Dandelion++

Improvements to Dandelion by Fanti et al.



- 4-regular anonymity graph instead of line-graph
- Messages flow through intertwined paths (cables)
- Epochs of 10 minutes
- Refreshes anonymity graph in each epoch

Why epochs?

Why epochs?

Case 1: Anonymity graph is always constant

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually
- Probability of successful mapping ~ 1

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually
- Probability of successful mapping ~ 1
- Prone to “graph learning attacks”

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually
- Probability of successful mapping ~ 1
- Prone to “graph learning attacks”

Case 2: Anonymity graph refreshed too frequently

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually
- Probability of successful mapping ~ 1
- Prone to "graph learning attacks"

Case 2: Anonymity graph refreshed too frequently

- Attackers use probabilistic similarity measures

Why epochs?

Case 1: Anonymity graph is always constant

- Attackers learn the graph eventually
- Probability of successful mapping ~ 1
- Prone to “graph learning attacks”

Case 2: Anonymity graph refreshed too frequently

- Attackers use probabilistic similarity measures
- Prone to “intersection attacks”

Why epochs?

Solution: Choose a time interval such that,

Why epochs?

Solution: Choose a time interval such that,

- Anonymity graph is constant within the interval

Why epochs?

Solution: Choose a time interval such that,

- Anonymity graph is constant within the interval
    - Attacker's knowledge of the network cannot be used in a probabilistic way

Why epochs?

Solution: Choose a time interval such that,

- Anonymity graph is constant within the interval
    - Attacker’s knowledge of the network cannot be used in a probabilistic way

- Network changes “enough” by the end of the interval

Why epochs?

Solution: Choose a time interval such that,

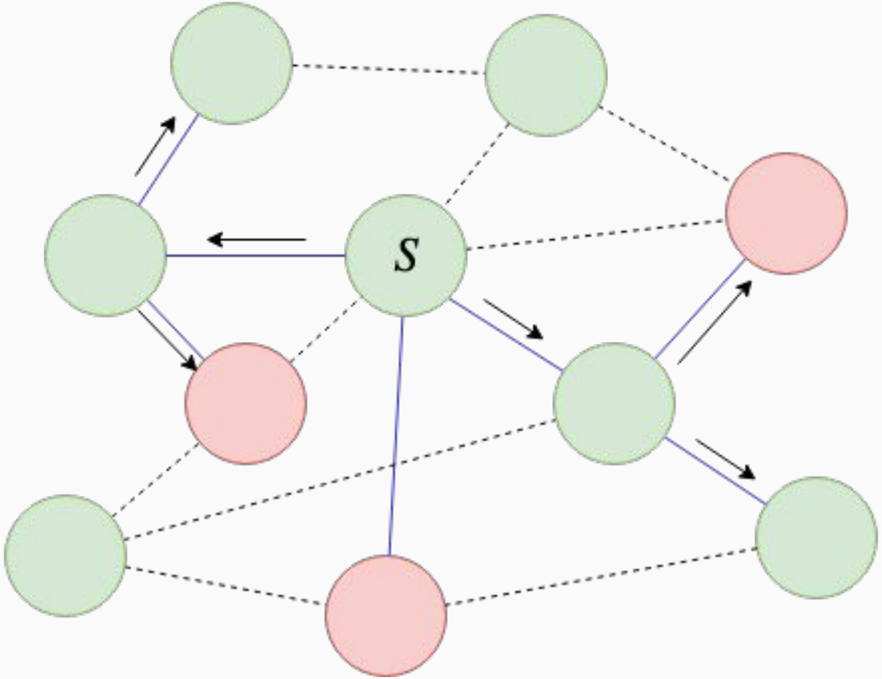
- Anonymity graph is constant within the interval
    - Attacker’s knowledge of the network cannot be used in a probabilistic way
- Network changes “enough” by the end of the interval
    - Attacker’s knowledge of the network is rendered useless

Synchronized Diffusion

- Aims to achieve asymmetric spreading, high propagation speed
- Operates in two phases:
  - Sync -- spreads transaction to relatively small “anonymity set”
  - Diffuse -- each node in anonymity set begins diffusion

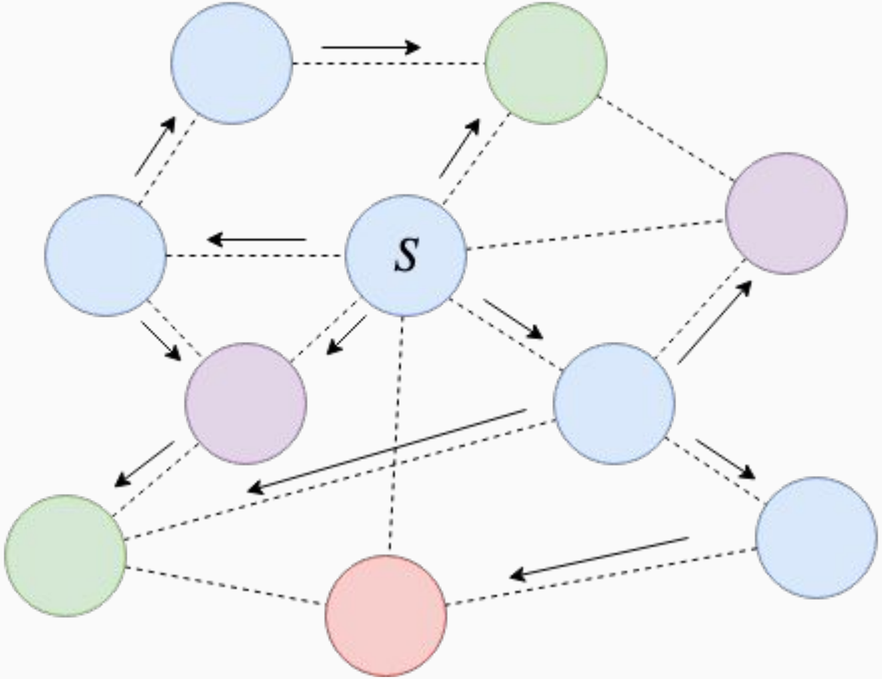
Synchronized Diffusion - Sync Phase

1. Message creator chooses time T_X
2. Send (X, T_X) to *m* neighbours
   - Always same *m* neighbours
3. If $T_{now} > T_X$: enter diffusion phase



Synchronized Diffusion - Diffusion Phase

- Run existing diffusion algorithm
- Idea: Several “simultaneous” starting points for diffusion
  - Enhances privacy
  - Fast propagation speed



Synchronized Diffusion - Extension

- Use epochs of 10 minutes
    - Hides anonymity graph when there are multiple transactions
- Use different *m* neighbours each epoch
    - Shorter epochs may leak more topology information
    - Longer epochs make transaction paths more predictable

Random Diffusion

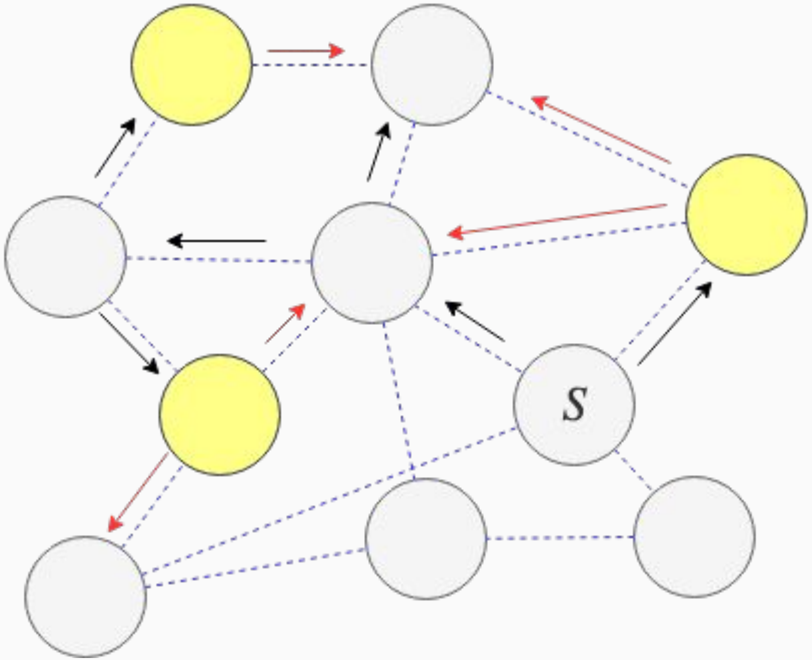
- Merges behaviour of previous protocols into single phase
- Two possible node behaviours
    - Relayers
    - Diffusers

Random Diffusion

Given some probability of diffusion, p_d

1. Compute H based on X and own secret key
2. If $H < p_d$: Run Diffusion

   Otherwise: Relay to *m* neighbours to continue Random Diffusion



Attacks

- Related literature
    - Learning the network topology
    - Observing the path taken by a message
    - Lokhov et al. on estimating the origin of epidemics
- Goal
    - Observe timestamps and sources of messages received
    - Estimate a mapping of transactions to nodes (IP, port)

Adversary

- Fraction *p* of all nodes are colluding spies
- Spies exchange information on a separate network external to Bitcoin
- Exact timestamps observed for transactions received at a spy node
- Types of attacks
    - Intersection attacks
    - Black hole attacks
    - Partial deployment attacks

Intersection Attack

- Exploits assumption: single transaction in an epoch
- Phase 1: Training
  - Learn the probability distribution vector P_v for each honest node v
  - $P_v(u)$ = Probability that node u is the first spy to hear from v
- Phase 2: Testing
  - Observe multiple transactions in an epoch
  - Construct the probability distribution vector Q
  - Find P_v that is closest to Q

Intersection Attack

$$C = \frac{P_v \cdot Q}{||P_v|| \, ||Q||}$$

- Cosine similarity used for comparing vectors
- Find the node for which similarity is maximum
- Map the set of transactions to node *v*

Black Hole Attack

- Stall the network instead of de-anonymize
- Transactions are not forwarded
- Solutions
    - Forward to more than one nodes
    - Time based fallback mechanism

Evaluation

- Evaluated 3 primary areas:
  - Resilience to intersection attacks
  - Propagation Speed
  - Partial Deployment

Evaluation - Intersection Attack

- Performed attacks with variety of parameters (# nodes, training size)
- Measured average precision and recall of adversary

Evaluation - Intersection Attack

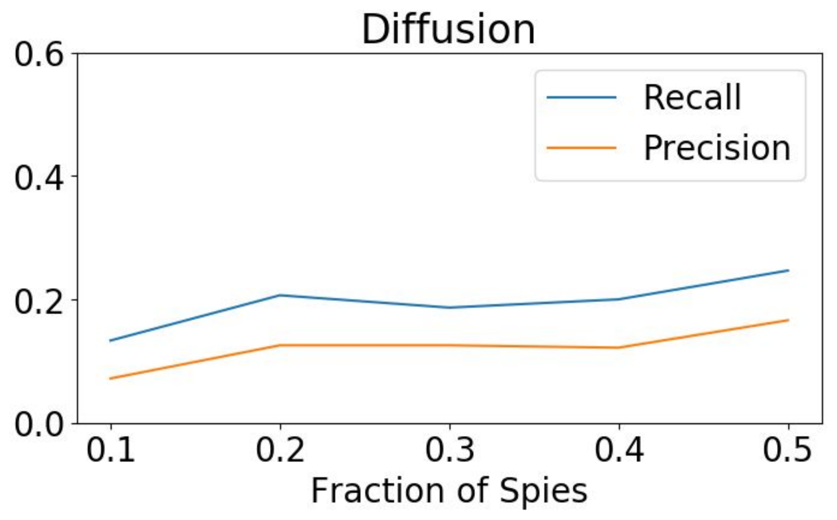
- Precision

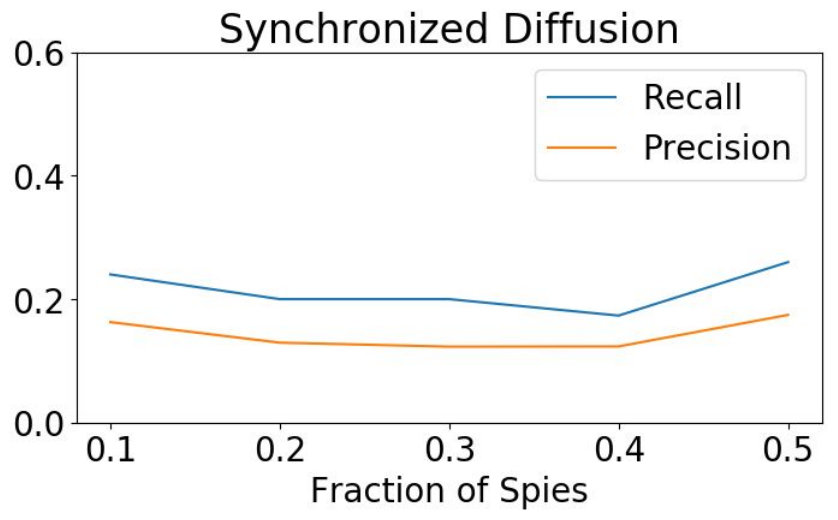
$$\frac{\text{True Positives}}{\text{TP + FP}} = \frac{\text{\# instances adversary correctly guesses source } v}{\text{Total \# times adversary guesses } v \text{ is source}}$$

- Recall

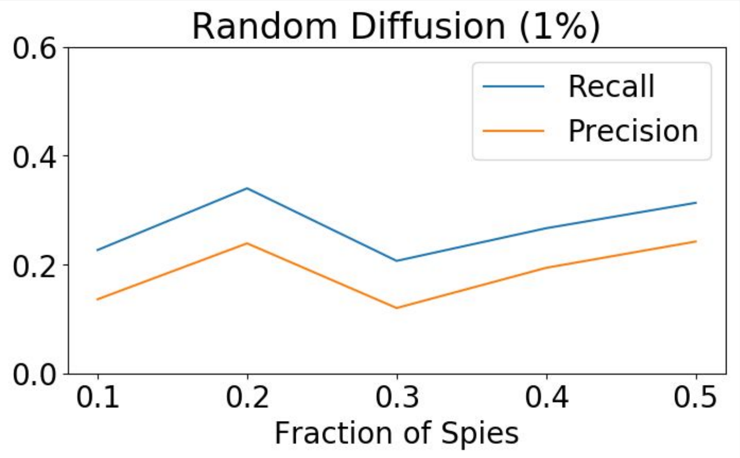
$$\frac{\text{True Positives}}{\text{TP + FN}} = \text{\% of sources correctly identified by adversary}$$

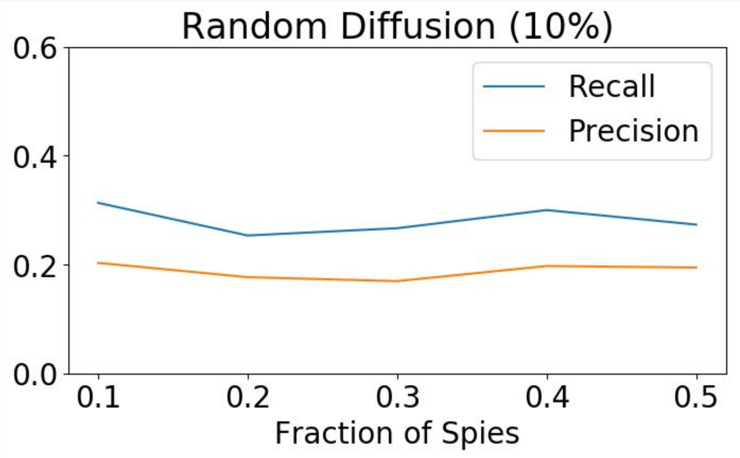
Evaluation - Intersection Attack

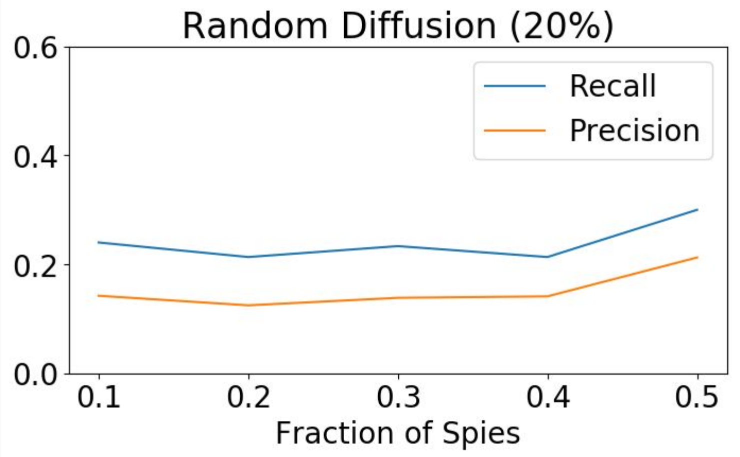




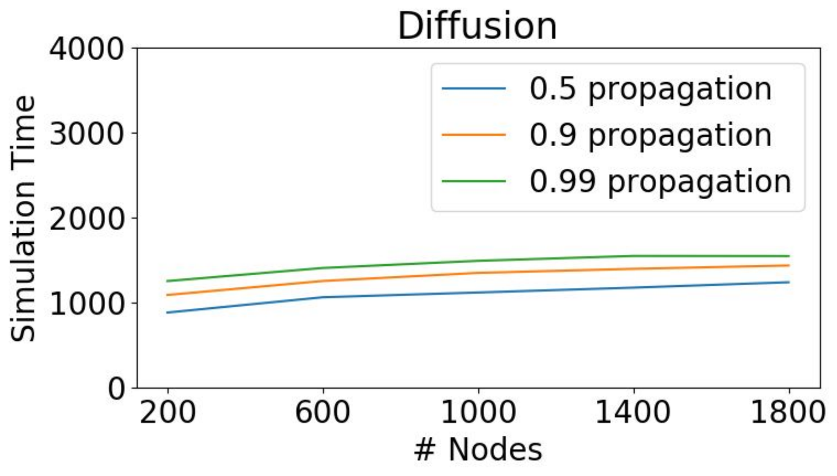
Evaluation - Intersection Attack

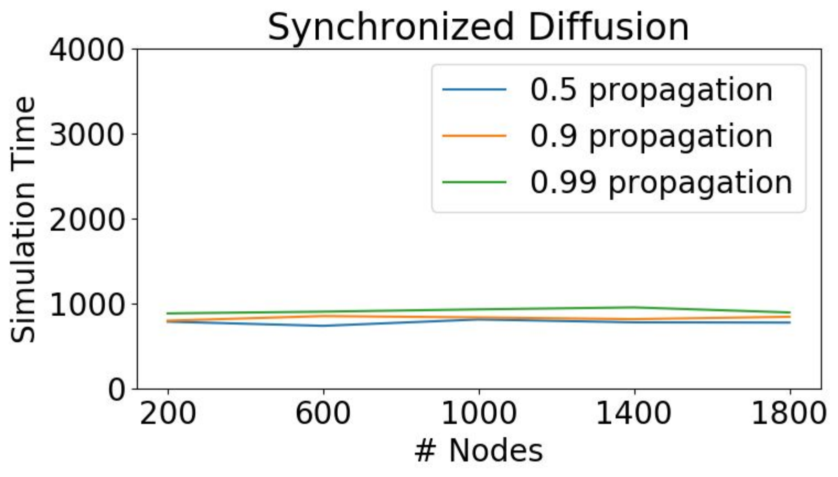




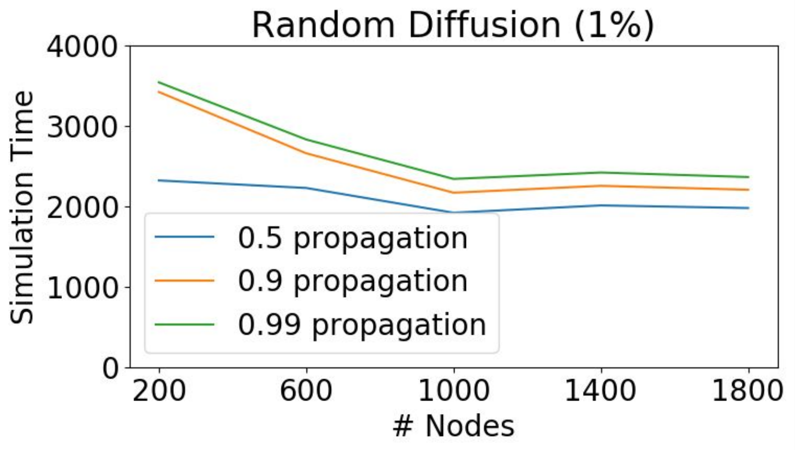


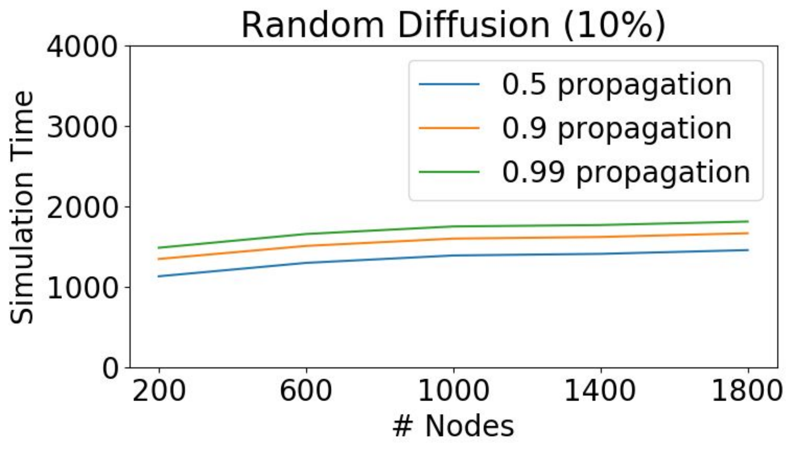
Evaluation - Propagation Speed

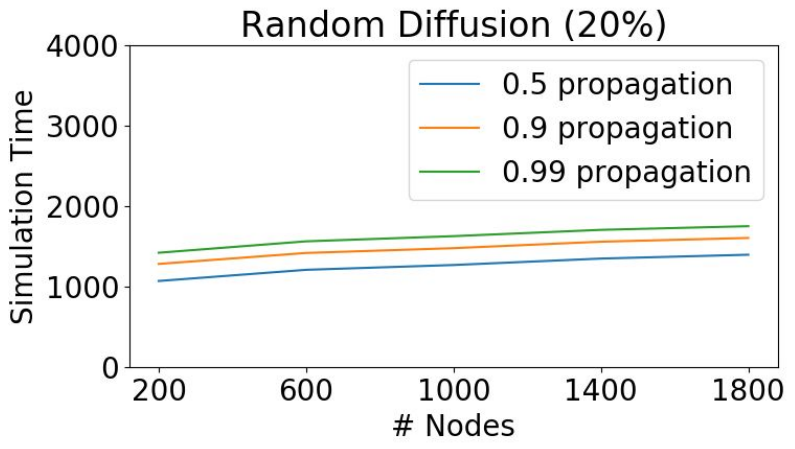




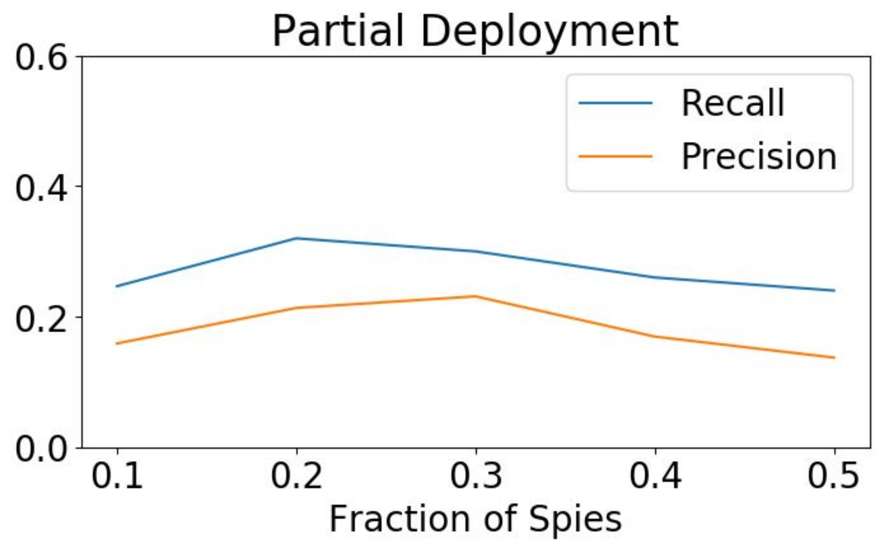
Evaluation - Propagation Speed

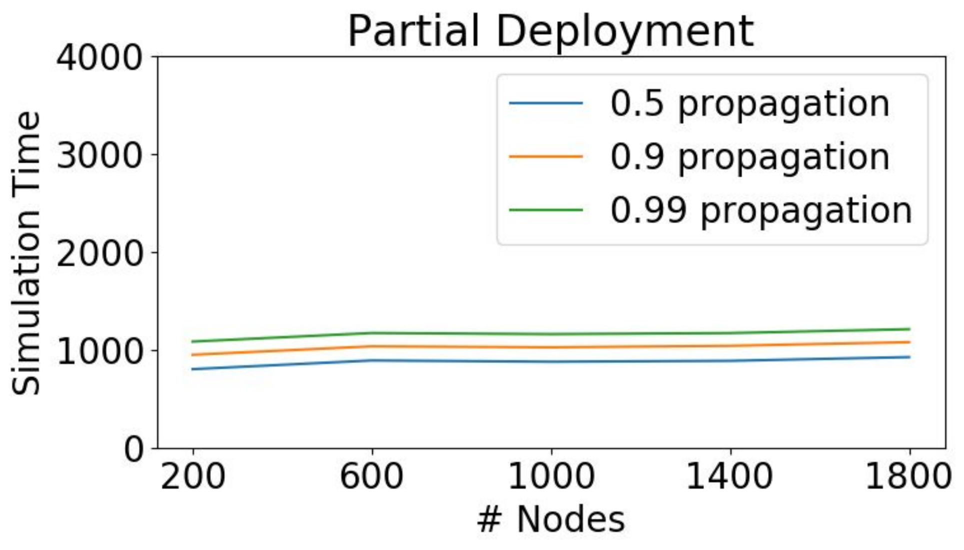






Evaluation - Partial Deployment





Conclusions

- Developed two Bitcoin spreading protocols
  - Sync-Diffusion
  - Random Diffusion
- Early results show reasonable resistance to intersection attacks
- Showed that partial deployment has small effect on speed/security

Future Work

- Compare with Dandelion++
- Provide theoretical analysis
- Test additional attacks