Bitcoin Boomerang

Protocol-Level Collaborative Anonymity for Transaction Broadcasting

Christopher A. Wood
Department of Computer Science
University of California Irvine
Email: woodc1@uci.edu

Chris H. Vu
Department of Computer Science
University of California Irvine
Email: ChrisHVu@gmail.com

***Abstract*—TODO**

I. Introduction

Bitcoin is a relatively young form of digital currency that has become increasingly popular in recent years. Its decentralized and peer-to-peer nature, in addition to its financial transaction security guarantees, makes it very appealing to alternative systems. However, providing adequate user anonymity when making transactions remains an open problem that has motivated a significant body of work studying alternative constructions, protocol modifications, and general studies on Bitcoin anonymity. Of the most prominent deanonymization techniques are network- and protocol-level analysis. Kaminsky [5] pioneered network-level attacks on client anonymity by eavesdropping on network activity and associating the flow of transactions with the IP addresses from which they originated. This type of attack served to circumvent the now standard practice of generating fresh key pairs and shadow change (output collection) addresses for every transaction so as to ensure unlinkability among users and their transactions.

Protocol-level studies and attacks, such as those performed by TODO, rely on the flow of transactions and other side-channel information in order to deanonymize clients. Even with the usage of anonymity layers such as Tor to hide their original IP addresses, such graph-based techniques are highly effective at linking transactions to their original users. Using eWallets or mixing services to further break the link between users and their transactions is often recommended, but there are numerous problems associated with these partial solutions. Perhaps most importantly, such services need to be trusted to not disclose the identity of their clients or simply steal a users funds. Designing mixing trustworthy and accountable mixing services has been studied several times in the literature, and the solutions to date either require modifications to the Bitcoin protocol [**?**] or have not yet been deployed to be adequately assessed in practice [**?**].

TODO: coinjoin/sharecoin

To circumvent the use of mixing services, others have proposed enhancements to the Bitcoin protocl that provide cryptographic guarantees of anonymity. Of these proposals, Zerocoin [7] appears to be the most feasible solution to implement in practice and see widespread adoption since it builds upon Bitcoin as a backing currency. Although there have been claims that its performance, which is based on RSA accumulators and zero-knowledge proof systems with expensive generation, verification, and large proofs, will ultimately impeded its acceptance, enhancements such as PinocchioCoin [**?**] have been proposed to allieviate such issues. PinocchioCoin provides the same functionality as Zerocoin but uses more efficient pairing-based primitives and the Pinocchio verificable computation scheme to replace expensive zero-knowledge proofs. Also, while briefly discussed in their respective publications, each of these "anonymous" coin schemes that build upon Bitcoin require a network-layer anonymizing layer such as Tor to prevent network-level attacks.

Based on these observations, it is clear that client anonymity guarantees at the network layer is fundamental for client anonymity at the level of the Bitcoin protocol. However, it would ideal if clients did not have to install or rely upon separate networking software to achieve this anonymity. To this end, we propose Bitcoin Boomerang, a peer-to-peer mixnet *built into* the Bitcoin protocol to provide virtually the same anonymity guarantees as network-layer services such as Tor without the same vulnerabilities (e.g., timing side channel attacks). Boomerang is an overlay network on top of the Bitcoin peer-to-peer (P2P) network. All Boomerang nodes must be connected to the Bitcoin P2P network as all nodes must be able to announce transactions. Due to this requirement, the design of Boomerangs network leverages the existing Bitcoin network to discover and connect to Boomerang nodes.

In this document we describe the detail the design of Bitcoin Boomerang, which shall henceforth be referred to as Boomerang for simplicity, and discuss the anonymity properties, expected performance, and preliminary implementation and simulation results that support our claims. The remainder of this document is outlined as follows. Section 2 provides an overview of the scheme, section 3 provides a detailed discussion of the Boomerand design, section 4 presents an analysis of the security and anonymity that Boomerang enables, and finally, section 5 discusses the expected performance of Bitcoin when using Boomerang for anonymity.

II. Review of the Bitcoin Peer-to-Peer Network

Bitcoin uses a simple broadcast network. To connect to the network, a local client uses four main methods of bootstrapping to locate an initial remote node. The first is the internal database of addresses the client saved during the previous session. If none of those addresses are active or the client is being run for the first time, the client uses a hard-coded DNS service to locate the address of seed nodes. The third method is via hard-coded addresses of seed nodes. The final

method uses user inputted addresses from the command line or loaded from a text file. The Bitcoin network also previously used Internet Relay Chat (IRC) bootstrapping, but support for this method has been removed as of Bitcoin version 0.8.2.

To connect to a remote node, the client node first sends a version message to the remote node. If the remote node accepts the version message it replies with a verack message and its own version message. If the client node accepts the remote nodes version message, it replies with a verack message. Finally, both nodes exchange getaddr messages and addr messages, which include the nodes address information as well as no more than 2,500 addresses seen in the last 3 hours in the nodes internal address database.

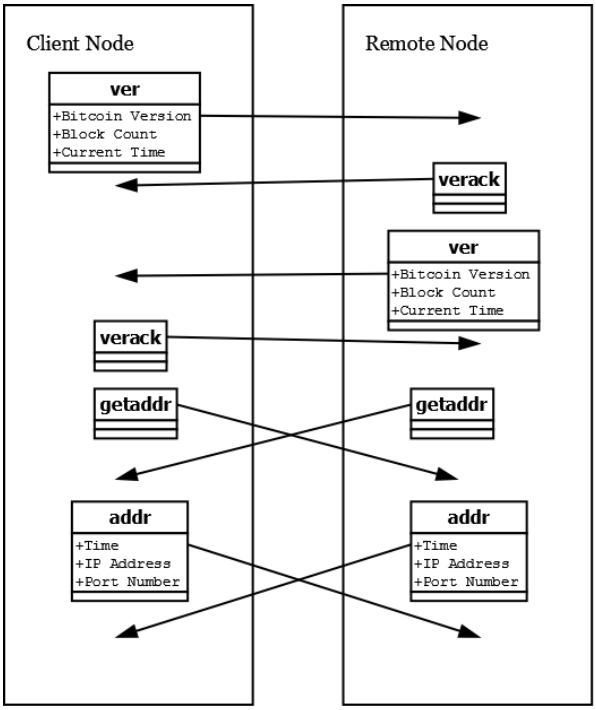

Fig. 1. Bitcoin connection setup.

After a node has joined the Bitcoin P2P network, peer discovery is propagated by callback addresses, addr messages relays, and self broadcast. Callback addresses are addresses in the remote nodes version message that enable the local node to connect back. If the local node does connect back, then as described above, the local node will exchange addr messages. Address relays occur after new addresses are added into the local nodes internal database. The local node will pick two random nodes in its database and relay the new addresses in another addr message. A node will also self broadcast, in which the node advertises its own address in an addr message to all connected nodes.

III. Boomerang Overview

At its heart, Boomerang is a peer-to-peer mixnet that can be leveraged by any anonymizing coin, such as Zerocoin, to provide anonymity analogous to Tor without the same side channel vulnerabilities. The main idea is to hide the source or origin from which transactions are introduced into the network. To do this, we integrate traditional mix networks into the peer-to-peer Bitcoin network such that new transactions which are to be broadcasted must first pass through a series of mixes (which are actually other participating Bitcoin users) through the network so as to obfuscate the originating network address. Functionally, this is no different from Tor. However, Boomerang has several important distinctions that make it unique with respect to onion-routing techniques like Tor:

1) Transaction anonymity increases *for every participant* when more people use Boomerang messages to broadcast transactions - everyone is therefore incentivized to use Boomerang.
2) Involuntary mixing delays mitigate timing-based side channel attacks that can be leveraged to deanonymize clients using Tor.
3) Boomerang messages are enhancements to the Bitcoin protocol, rather than a means for anonymizing point-to-point TCP connections, as is done with Tor.

The main design goals of Boomerang are as follows:

Sender anonymity
: A node achieves *sender anonymity* if a node cannot be (uniquely) identified, or linked, to a particular transaction that is broadcast at the end of a Boomerang circuit (see Section IV) [4].

Fault-tolerance via redundancy
: Compromised or mobile nodes should not lead to significant delays in transaction introduction or compromise mixnet deadlocks.

Performance
: Nodes in the network should incur minimal overhead from using Boomerang messages while maximizing their anonymity.

We now provide a brief overview of how Boomerang is used by clients for transaction broadcast anonymity; A complete description of the protocol is provided in Section 4. To broadcast a new transaction T using Boomerang, a client C first creates a set of W individual mixing circuits of length $D_1, D_2, \ldots, D_W$, where $W \geq 1$ and $D_i \geq 2$. Let $K = \sum_{i=1}^{W} D_i$ be the total number of nodes selected as mixing services for the circuits. With knowledge of the public keys for each of the K nodes, C then wraps T in D_i layers of encryption for each circuit $i = 1, \ldots, W$. Each layer of encryption also includes a forward pointer to the subsequent hop in the circuit so that decrypting nodes may forward the transaction to the appropriate location, or broadcast the plaintext transaction T if they are the last hop in the circuit.

As is standard with traditional mix networks [2], each nodes will only forward wrapped transactions after it has accumulated a certain number of transactions from other nodes. For this reason, to avoid network deadlock, we require that "cover traffic" be circulated throughout the network to keep traffic moving smoothly, similar to the cover traffic used in the Tarzan mix network [3]. Furthermore, this cover traffic must encoded such that it appears indistinguishable from legitimate encrypted transaction messages. To solve this problem, Boomerang cover traffic messages *are legitimate transaction encryptions*, with the exception that the last "hop" of the "circuit" for these messages is the same client C that generated them in the first

place. Thus, these dummy transaction messages will be circulated throughout the network and ultimately be routed to C, who can then easily check that they generated the transaction and discard it (or send out another dummy message). The circular nature of this cover traffic, which is shown by the red trajectories in Figure 2, is the inspiration for Boomerang's name.

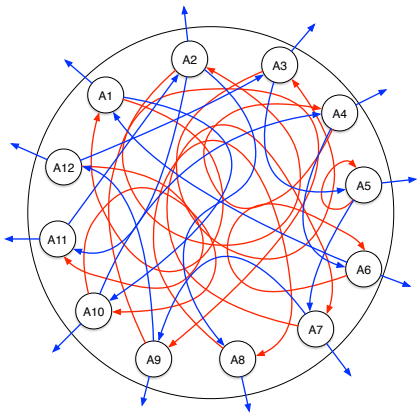

Fig. 2. TODO

Given this short description, there are many important engineering problems to solve in order make Boomerang feasible in practice, such as public key distribution and usage, prevention of self-induced denial-of-service attacks, and the overconsumption of network resources. We describe solutions to all such problems in Section 4, and include a preliminary performance analysis of the Boomerang technique (based on analytical modeling and simulations) in Section 5.

IV. Boomerang Design

As previously discussed, the Boomerang protocol enhancement for Bitcoin is motivated by the need to hide the source from which transactions are introduced into the network. Furthermore, this should be done in a transparent way so that any other form of anonymous coin extension on top of Bitcoin (e.g., Zerocoin) can leverage the service for transaction anonymity. Boomerang is *not* intended to support regular Bitcoin traffic; once a transaction becomes public knowledge, Boomerang no longer plays a role in its distribution.

In the following sections we detail the core protocol and several important design and security tradeoffs that can be made in practice when using Boomerang. A formal analysis of the security and performance of Boomerang-enhanced Bitcoin is provided in Sections X and Y.

A. Boomerang Peer-to-Peer Network Design

Boomerang peer discovery is done in a similar manner to the Bitcoin peer discovery process described above with a few exceptions. Boomerang nodes can also query Bitcoin nodes after connecting to the Bitcoin network. Boomerang nodes can also learn the address of other nodes by routing Boomerang messages. The Boomerang peer discovery methods are:

1) User inputted addresses from command line or text file: If the user manually inputs address information, the client will attempt to connect to those Boomerang peers first.
2) Node reads stored Boomerang addresses from previous sessions: Boomerang clients will attempt to connect to peers stored in the internal address database from the previous session.
3) Nodes make a DNS request for Boomerang seed nodes: If there are no active nodes in the internal address database, then the client will perform a DNS lookup using hard-coded DNS servers for current seed nodes.
4) Nodes connect to hard-coded seed addresses: If DNS lookups fail, then the client attempts to connect to hard-coded seed addresses.
5) Nodes query Bitcoin network for Boomerang nodes: The client sends Boomerang connection requests to nodes on the Bitcoin network.
6) Nodes utilize callback addresses: After receiving a Boomerang connection request from a remote node, the client node can use the callback address to connect to the remote node and exchange address database information.
7) Nodes receive relayed addresses: After receiving new address information and verifying validity of the address, a node will randomly pick a few nodes in the internal database and relay the new address information.
8) Nodes will self broadcast periodically: Similarly to relayed addresses, periodically Boomerang clients will randomly pick a few nodes in the internal database and relay its own address information.
9) Boomerang messages: The destination of a Boomerang message must be a Boomerang node. The client can then request a connection with the destination node and exchange address database information.

B. Broadcast Protocol

At the heart of the Bitcoin protocol is the ability to encode new transactions as Boomerang messages and then ripple them throughout the network. We describe the complete procedure for message encoding, EncodeTransaction, in Algorithm 1, where the notation contained therein is defined in Table **??**. An encoded Boomerang message has a very well-defined format, as shown in Figure 3. In particular, the message is composed of the following:

1) A potentially re-encrypted seed. By the description of EncodeTransaction, it is required that the public-key encryption scheme used to mask these seeds has the same domain and range. This is needed because the decrypted seed for one hop will be used as decrypted seed on the previous hop, very much like onion layers of encryption.
2) An encrypted address vector that is used by each hop to learn the next hop in the circuit without learning any other information about the nodes in the circuit. More specifically, a router can only learn about the immediate source and destination of a Boomerang message (the security and anonymity implications of this are discussed in the following section).

3) A potentially re-encrypted transaction message block. This block either stores the encrypted transcaction, where the encryption is done by XORing with a pseudorandom bit string generated by the decrypted seed value, or the plaintext transaction that is to be broadcast throughout the network.

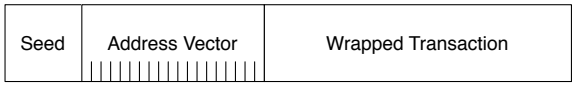

Fig. 3. Boomerang message encoding.

The procedure to handle Boomerang messages, BoomerangMessageHandler, is provided in Algorithm 2.

Similar to the Tarzan P2P mixnet, a critical part of the Boomerang protocol is the inclusion of cover traffic that is indistinguishable from legitimate encoded Boomerang transaction messages [3]. This traffic is needed for two reasons: (1) to keep legitimate transactions moving through mixnet circuits, and (2) to obfuscate the flow of legitimate transactions through the network. To support this cover traffic with minimal changes to the protocol, nodes in the network will *reuse* and *re-encode* old transactions to be sent throughout the network, with the exception that the destination node for the mixnet circuit (as specified in the EncodeTransaction procedure) will be the same as the sender. This is because the sender can easily discover when a transaction message has looped through the network and back to themselves, at which point they can then simply discard the transaction. Clearly, the rate at which this cover traffic is generated plays a critical role in the overall performance of the system when using Boomerang. We discuss the selection of parameters that achieve optimal performance without sacrificing security in Section 5.

C. Cryptographic Primitives

Based on the EncodeTransaction and BoomerangMessageHandler procedures, we require the following cryptographic primitives to support Boomerang messages:

1) Uniform domain and range public-key encryption without ciphertext expansion, and
2) Deterministic PRG whose input is an element in the range specified by the public-key encryption scheme.

To satisfy the first primitive, we chose the standard RSA cryptosystem. While ElGamal has the ideal property that plaintext and ciphertext elements are members of the same group $\mathbb{Z}_p$ for some sufficiently large prime p, the ciphertext actually consists of a pair of group elements, and thus does not satisfy the ciphertext expansion property. Also, while RSA-OAEP is the only known CCA-secure variant of RSA (in the Random Oracle model), the input plaintext is strictly less than the output ciphertext due to the applied padding. Therefore, this does not achieve the first property that function domain and range are equivalent.

D. Node Verification

Boomerang clients need to validate addresses in the internal database. In order to validate addresses, Boomerang messages will be sent through un-validated addresses as well as validated addresses. If the local client correctly decrypts the Boomerang message, the nodes along the messages route are marked as verified with a timestamp in the internal database. Boomerang clients will only relay address information from verified addresses.

V. Boomerang Properties

In this section we elaborate on the design goals of Boomerang. We first discuss the anonymity properties that are provided by Boomerang, and then discuss system-level properties such as fault tolerance and performance.

A. Anonymity Properties

Inspired by Tarzan [3], we analyze the anonymity properties of Boomerang with respect to static and adaptive adversaries. In particular, we strive to show that senders achieve anonymity against a minority of colluding nodes. Before proving any claims, we present the main sources of information exposure in Table II; a positive entry indicates that an attacker will be able to uncover the source of information, whereas a negative entry indicates that such exposure is not feasible given the Boomerang design.

B. System-Level Properties

From a systems perspective, Boomerang was designed with fault-tolerance and performance in mind. In particular, we (easily) claim that the Boomerang scheme is resistant to any adversarial attempt to leverage a successful denial of service (DoS) attack on the network. By the assumptions of the Bitcoin network, a majority of the participating nodes will always be honest (i.e., effectively uncompromised). Now, assume that there are N_{total} total nodes in the Bitcoin network, at least $N_{total}/2$ such nodes are honest, and $N_{bad} \leq T/2 - 1$ nodes are compromised. By this claim, during node selection and circuit formation, W nodes will be drawn at random without replacement, meaning that the probability of forming a circuit with at least one corrupt node, in the worst case, is

$$\sum_{i=1}^{D} \frac{N_{bad}}{(N_{total}/2) - 1 - i}$$

For reasonable measures of fault-tolerance, we require that this probability is kept as small as possible. However, for anonymity purposes, we require that D is maximized to achieve optimal mixing (and thus, anonymity) throughout the network.

need to quantify anonymity in terms of N and T.

From a performance perspective, our main goal is to minimize the overhead of Boomerang message encoding, transmission, and forwarding while maximizing the anonymity properties discussed in the previous section. To that end, we discuss how the performance varies based on system-wide parameters that influence such anonymity properties. Based on the Boomerang design it is clear that the performance of the Boomerang scheme is tightly coupled to the following paramters:

TABLE I. BOOMERANG PROTOCOL NOTATION

Symbol	Description

Algorithm 1 EncodeTransaction(T)

for $i = 1$ to W **do**
  $\bar{T} := T$
  $s \leftarrow \{0,1\}^{\tau}$
  for $j = 1$ **to** D_i **do**
    $p := \mathsf{PRG}(s)$
    $\bar{T} := \bar{T} \oplus p$
    $s \leftarrow E_{pk_{i,j}}(s)$
  end for
  $\mathsf{index} \leftarrow \{0, \ldots, 2D_i\}$
  $\mathsf{AV} := [2D_i]$
  for $j = D_i$ **downto** 2 **do**
    $\mathsf{AV}[\mathsf{index}] := E_{pk_{i,j}}(\mathsf{addr}_{D_j})$
    $\mathsf{index} := \mathsf{index} + 1(\mod N_m)$
    $\mathsf{AV}[\mathsf{index}] := E_{pk_{i,j}}(\mathsf{addr}_{D_{j-1}})$
    $\mathsf{index} :=\leftarrow \{0, \ldots, 2D_i\}$
    while $\mathsf{index} \mod 2 \neq 0$ and $\mathsf{AV}[\mathsf{index}] \neq \perp$ and $\mathsf{AV}[\mathsf{index} + 1(\mod \mathsf{D_i})] \neq \perp$ **do**
      $\mathsf{index} \leftarrow \{0, \ldots, 2D_i\}$
    end while
  end for
  $M := \mathsf{Pack}(s, \mathsf{AV}, \bar{T})$
  $\mathsf{Transmit}(M)$
end for

Algorithm 2 BoomerangMessageHandler(j, M)

$s := D_{sk_j}(M[0])$
$\bar{T} := M[2] \oplus PRG(s)$
if $\bar{T}$ is a well formed transaction **then**
  $\mathsf{Broadcast}(\bar{T})$ to the Bitcoin network
else if $\bar{T}$ destination address is addr_j **then**
  Discard $\bar{T}$; return;
else
  $\mathsf{AV} := M[1]$
  $i := 1$
  while $i < \mathsf{len}(\mathsf{AV})$ **do**
    $\mathsf{addr}_{src} := D_{pk_j}(AV[j])$
    if $\mathsf{addr}_{src} = \mathsf{addr}_j$ **then**
      $\mathsf{addr}_{dst} := D_{pk_j}(AV[j+1])$
      $M := \mathsf{Pack}(s, \mathsf{AV}, \bar{T})$
      if $|Buffer| \geq B$ **then**
        $\mathsf{Transmit}(M)$
      else
        $B.add(M)$
      end if
    else
      $i := i + 2$
    end if
  end while
end if

- W - the circuit width,
- D - the circuit depth,
- σ - the average cover traffic generation rate, and
- π - the average rate at which new transactions are made.

TABLE II. BOOMERANG INFORMATION EXPOSURE.

Information Exposed	*Bad Entrance Node*	*Bad Intermediate Node*	*Bad Exit Node*	*Bad Entrance/Exit Nodes*
Sender activity	Maybe	Maybe	No	Maybe
Sender content	No	No	No	Maybe

How many messages should a node expect to receive in a set amount of time?

C. Network Security Analysis

Analysis assumes a clients first connection is to an honest node. If the client first connects to a malicious node, no security is possible as the malicious node will trivially keep the client in the malicious network. The adversary can use many long-lived identities to bias the distribution of nodes towards the malicious network. If the adversary controls a majority of nodes, no security guarantees are possible. Therefore, security analysis will assume a client node to be connected to some honest nodes and the adversary does not control the majority of nodes.

The adversary can introduce many invalid addresses into the network. By using Boomerang messages to validate addresses, honest nodes can aggressively prune invalid addresses from their internal databases. In a similar fashion, Boomerang messages will prune nodes that improperly modify data that is routed through them.

Due to the open nature of the Boomerang network, no solution to denial-of-service attacks are currently possible.

VI. PERFORMANCE ANALYSIS

TODO: from the simulation

A. Parameter Selection

VII. RELATED WORKS

Tarzans P2P network is much more secure than Boomerang, but may not function well in the Bitcoin environment due to scalability. Tarzans peer discovery protocol involves gossip and validation. Nodes gossip network information by exchanging all known addresses and corresponding public key hashes. A node validates an address by directly connecting to it with a gossip message that includes a nonce.

The benefit of Tarzans gossip procedure is that malicious nodes cannot respond with a different public key for every node in the network. Nodes learn public keys from gossip and not from the owner of the keys. Tarzans validation method prevents the propagation of invalid address information throughout the network. Assuming the first node a client connects to in the Tarzan network is honest, the client will eventually learn of every honest node in the network. The Tarzan network is not as scalable due to the amount of communication required to set up and maintain the network.

Peer-to-peer distributed hash table (DHT) networks, such as Chord and Kademlia, use a distributed system to keep track of key-value pairs. DHT networks use a 160-bit key space. The Chord network has the keys arranged in a circle. Each node in the Chord network maintains a segment of the circle adjacent to it. A successor to a Chord node is the node that

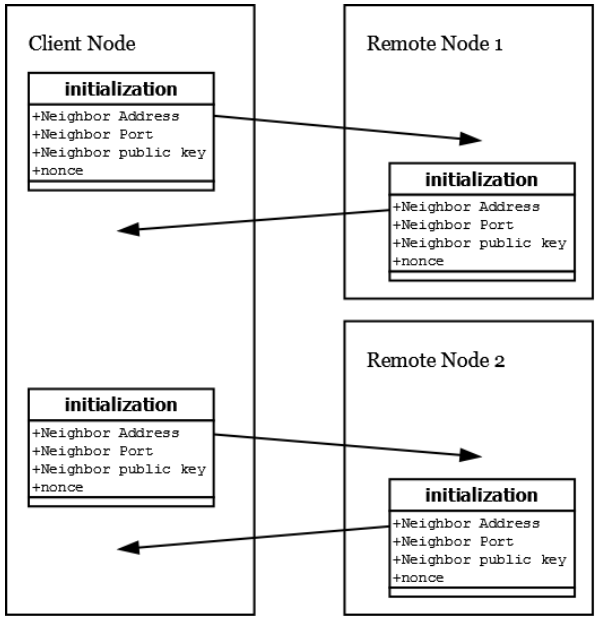

Fig. 4. Tarzan gossip and validation protocol.

maintains the next set of keys. By searching linearly through nodes, the node responsible for a key-pair can be found. To speed up searches, each node also keeps a finger table that contains successor information as shown below.

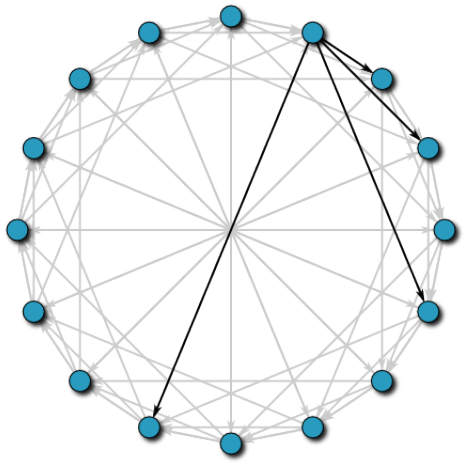

Fig. 5. A Chord network with "finger" table entries highlighted.

Kademlia DHT is similar to Chord except that it is organized as a tree and distance is calculated using the exclusive or operation. The exclusive or operation allows the nodes in the Kademlia network to have symmetric distances between two

nodes. Kademlias simpler distance calculation and iterative search, which is made possible by the tree-shaped network, allows for much faster searches than the Chord network. Both networks are decentralized, autonomous network that has fault tolerance and scalability. Unlike the Tarzan network, DHT networks do not prevent attacks on key searches or index poisoning and therefore allow a few malicious nodes to prevent nodes from finding other honest nodes.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *Consulted* 1 (2008).

[2] David L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM* 24(2) (1981), 84-90.

[3] Michael J. Freedman and Robert Morris. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM conference on Computer and Communications Security*, ACM (2002).

[4] A. Pfitzmann AND M. Køohntopp. Anonymity, Unobservability and Pseudonymity A Proposal for Terminology. *In Federrath* 12, 1-9.

[5] Dan Kaminsky. Black Ops of TCP/IP Presentation. *Black Hat, Chaos Communication Camp* (2011).

[6] David L. Chaum. Blind Signatures for Untraceable Payments. *Crypto* 82 (1982).

[7] Ian Miers, Christina Garman, Matthew Green, Aviel D. Rubin. Zerocoin: Anonymous Distributed E-Cash from Bitcoin. *IEEE Symposium on Security and Privacy* (2013).

[8] Tor. Bitcoin Wiki. Available online at https://en.bitcoin.it/wiki/Tor. Last accessed: 1/25/14.