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 [**?**] 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 [**?**] 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).

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 guarantees, 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. 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.

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 selects a set of M individual mixing circuits of length $N_1, N_2, \ldots, N_M$, where $M \geq 1$ and $N_i \geq 2$. Let $K = \sum_{i=1}^{M} N_i$ 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 N_i layers of encryption for each circuit $i = 1, \ldots, M$. 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 1, is the inspiration for Boomerang's name.

Given this brief 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.

III. Design Goals

2. Anonymity against malicious nodes: Tarzan should provide sender or recipient anonymity against colluding nodes. That is, a particular host should not be uniquely linkable as the sender (recipient) of any message, or that a message should not be linkable to any sender (recipient) [20]. We consider these properties in terms of an anonymity set: the set of

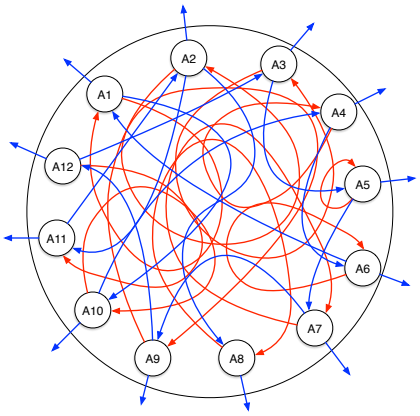

Fig. 1. TODO

possible senders of a message. The larger this set, the more anonymous an initiator remains. These properties implies the weaker relationship anonymity: an adversary should not be able to identify a pair of hosts as communicating with each other, irrespective of which host is running Tarzan.

3. Fault-tolerance and availability: Tarzan should resist an adversarys attempts to overload the entire system or to block system entry or exit points. Tarzan should minimize the damage any one adversary can cause by running a few compromised machines.

4. Performance: Tarzan should maximize the performance of tunnel transmission, subject to our anonymity requirements, to make Tarzan a viable IP-level communication channel.

5. Anonymity against a global eavesdropper: An adversary observing the entire network should be unable to determine which Tarzan relay initiates a particular message. Therefore, a nodes traffic patterns should be statistically independent of it originating data traffic.

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. 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 2. 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. 2. 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.

B. 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.

C. Peer Discovery

TODO: gossip protocol, duration of Boomerang public/private key pairs, what else?

V. Performance Analysis

TODO: from the simulation

A. Parameter Selection

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).

TABLE I. BOOMERANG PROTOCOL NOTATION

Symbol	Description

Algorithm 1 EncodeTransaction(T)

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

Algorithm 2 BoomerangMessageHandler(n, M)

$s := D_{sk_n}(M[0])$
$\bar{T} := M[2] \oplus PRG(s)$
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_n **then**
  Discard $\bar{T}$; return;
else
  $\mathsf{AV} := M[1]$
  $n := 1$
  while $n < 2N_m$ **do**
    $\mathsf{addr}_{src} := D_{pk_n}(AV[n])$
    if $\mathsf{addr}_{src} = \mathsf{addr}_n$ **then**
      $\mathsf{addr}_{dst} := D_{pk_n}(AV[n+1])$
      $M := \mathsf{Pack}(s, \mathsf{AV}, \bar{T})$
      if $|Buffer| \geq B$ **then**
        $\mathsf{Transmit}(M)$
      else
        $B.add(M)$
      end if
    else
      $n := n + 2$
    end if
  end while
end if