Utilizing Public Blockchains for the Sybil-Resistant Bootstrapping of Distributed Anonymity Services

Roman Matzutt, Jan Pennekamp, Erik Buchholz, Klaus Wehrle
{matzutt,pennekamp,buchholz,wehrle}@comsys.rwth-aachen.de
Communication and Distributed Systems, RWTH Aachen University, Germany

ABSTRACT

Distributed anonymity services, such as onion routing networks or cryptocurrency tumblers, promise privacy protection without trusted third parties. While the security of these services is often well-researched, security implications of their required bootstrapping processes are usually neglected: Users either jointly conduct the anonymization themselves, or they need to rely on a set of non-colluding privacy peers. However, the typically small number of privacy peers enable single adversaries to mimic distributed services. We thus present *AnonBoot*, a Sybil-resistant medium to securely bootstrap distributed anonymity services via public blockchains. AnonBoot enforces that peers periodically create a small proof of work to refresh their eligibility for providing secure anonymity services. A pseudo-random, locally replicable bootstrapping process using on-chain entropy then prevents biasing the election of eligible peers. Our evaluation using Bitcoin as AnonBoot's underlying blockchain shows its feasibility to maintain a trustworthy repository of 1000 peers with only a small storage footprint while supporting arbitrarily large user bases on top of most blockchains.

CCS CONCEPTS

• **Security and privacy** → **Pseudonymity, anonymity and untraceability**; • **Networks** → **Peer-to-peer protocols**.

KEYWORDS

anonymization; bootstrapping; public blockchain; Sybil attack; anonymity network; cryptocurrency tumbler; Bitcoin; Tor

ACM Reference Format:
Roman Matzutt, Jan Pennekamp, Erik Buchholz, Klaus Wehrle. 2020. Utilizing Public Blockchains for the Sybil-Resistant Bootstrapping of Distributed Anonymity Services. In *Proceedings of the 15th ACM Asia Conference on Computer and Communications Security (ASIA CCS '20), October 5–9, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3320269.3384729

1 INTRODUCTION

Preserving user privacy on the Internet has become a complex task due to increasingly pervasive measures for online surveillance: While re-establishing their anonymity traditionally was only crucial for a set of especially privacy-aware users, the Snowden revelations have shown that every online user's privacy is at stake [23]. This shift further fueled distributed anonymity services, such as message shuffling networks [13], anonymous communication networks based on onion routing [16], or cryptocurrency tumblers [31, 54, 55]. While various works have investigated secure building blocks for anonymity services, those works typically overlook the bootstrapping of such services. Often, related work simply assumes non-colluding peers, e.g., because of their operators' presumed real-world reputation. However, this perceived reputation does not always warrant trust, as evidenced, e.g., by numerous alleged scams regarding cryptocurrencies [2, 49, 55] and the need for manually reporting [45] or actively probing [12, 50] bad peers in the Tor network. Hence, the question remains: *How to securely bootstrap distributed anonymity services without having to rely on operator reputation?*

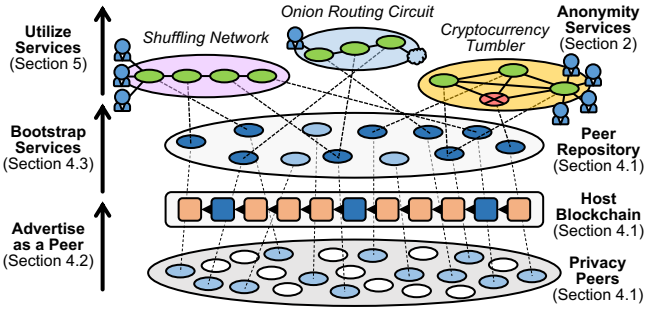
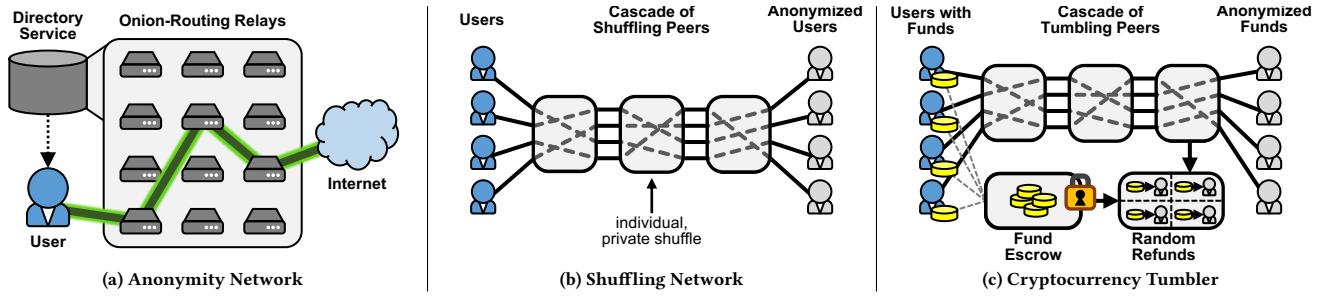


Figure 1: High-level design overview of AnonBoot, our medium for securely bootstrapping anonymity services.

In this paper, we propose to outsource privacy-enhancing tasks to small networks of peers selected randomly in a secure, unbiased, and transparent fashion from a Sybil-resistant peer repository. We introduce *AnonBoot* as a medium for indexing and bootstrapping these anonymity services on top of a public host blockchain, which provides accepted means to maintain an immutable and transparent event log. As we illustrate in Figure 1, peers join by periodically publishing advertisements containing a small proof of work (PoW) to the host blockchain. Peer operators thus need to periodically invest hardware resources into refreshing their membership within a limited time frame, and all participants can locally derive AnonBoot's state by monitoring the host blockchain. Hence, AnonBoot creates a Sybil-resistant index of privacy peers from which users can then request bootstrapping new anonymity services. Users can choose privacy peers or established anonymity services from this index to cater to their individual privacy requirements. We exemplarily build AnonBoot on top of Bitcoin to showcase its low requirements as well as the small storage footprint it has on its host blockchain, and to show that our system does not require sophisticated blockchain features, such as smart contracts, to operate.

ASIA CCS '20, October 5–9, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-6750-9/20/10…$15.00
https://doi.org/10.1145/3320269.3384729



(a) Anonymity Network (b) Shuffling Network (c) Cryptocurrency Tumbler

Figure 2: Well-known distributed anonymity services encompass (a) anonymity networks, such as Tor, for anonymous Internet communication, (b) message-shuffling networks, and (c) cryptocurrency tumblers to increase users' financial privacy.

Contributions.

- By analyzing existing anonymity services (Section 2), we identify a lack of secure bootstrapping for such services (Section 3).
- Through AnonBoot[1], we show that public blockchains are a suitable basis to create such a secure bootstrapping process (Section 4) for heterogeneous established use cases (Section 5).
- We show that PoW and peer election can prevent adversaries from gaining advantages over honest peer operators (Section 6).
- AnonBoot scales to repositories of, e.g., 1000 privacy peers and large user bases with only low storage impact on its host blockchain and low, tunable costs for its participants (Section 7).

2 AVAILABLE ANONYMITY SERVICES

We identify three categories of distributed anonymity services for outsourcing privacy management: Internet anonymity networks, message shuffling networks, and cryptocurrency tumblers.

2.1 Anonymity Networks

Anonymity networks, such as Tor [16], enable low-latency and anonymous Internet communication through onion routing, i.e., tunneling users' traffic through a user-selected circuit under a layered encryption, as we exemplify in Figure 2a. The user creates her circuits locally at random, but she also considers performance metrics, such as available bandwidth at individual nodes [15], as well as node-specific policies, e.g., exit nodes only performing requests to certain ports on the user's behalf [16]. Tor provides the information required to build circuits through a *directory* that is maintained by exceptionally trusted *directory servers* [16]. These, currently ten [44], directory servers are vetted by the Tor project maintainers, and users must trust that those directory servers do not collude [36]. To further increase the reliability of this directory, relays are actively being probed [12, 50], and users can report misbehavior to the Tor project [45]. Thus, misbehaving nodes are flagged in the directory to enable users to avoid such relays [45].
Takeaway. Tor relies on an index of available nodes and their properties but requires trusted authorities to maintain this index.

2.2 Message Shuffling Networks

Long before the recent proliferation of anonymity networks, David Chaum introduced networks for oblivious message shuffling [13], to which we refer to as *shuffling networks*, as a means to realize anonymous mail systems that provide sender anonymity, e.g., to protect whistleblowers from retribution. Figure 2b showcases the basic user interaction with such a shuffling network. Similarly to anonymity networks, users relay their messages through a cascade of known *shufflers*, again after encrypting them in layers. However, *multiple* users shuffle their messages through the *same* cascade of nodes to achieve a vastly reduced overhead. These shufflers hence, one after another, receive the batch of encrypted messages of which they can lift only the outermost encryption layer. After decrypting the message batch, each shuffler obliviously shuffles the batch's messages and forwards the result to the subsequent shuffler. Therefore, shufflers are unable to correlate other shufflers' input and output batches. As long as one shuffler remains honest, no passive adversary can deanonymize the users from now on.

However, shuffling networks are often prone to active attacks, such as denial of service (DoS) or replacing encrypted messages [14]. Furthermore, adversaries can easily operate full shuffling networks at low costs since those networks are fixed and small in size.
Takeaway. Users need to trust that non-colluding operators run the shuffling network faithfully, which is especially challenging due to the current lack of a widely accepted index of shuffling networks.

2.3 Cryptocurrency Tumblers

Multiple analyses of public blockchains, especially Bitcoin [32, 35, 38], debunked the initial hope that cryptocurrencies provide sufficient user privacy by not building up long-lived identities [34]. To counteract curious blockchain observers, *cryptocurrency tumblers*, or *cryptotumblers*, break the linkability of privacy-aware users and their funds. Cryptotumblers pool the funds of multiple users and then pay out random coins of the same value to each user such that the new coin owners are unknown to blockchain observers.

Cryptotumblers evolved over time, yielding different generations and flavors to appropriately address users' security and privacy concerns. First, users of centralized cryptotumblers require strong trust in the service operator to not steal their funds or disclose the shuffling history at a later point to deanonymize users. Series of alleged scams [2, 49, 55], however, underpin the need for further *technical protection*, e.g., holding the cryptotumbler accountable [11].

The first generation of distributed cryptotumblers let privacy-aware users jointly simulate a centralized tumbler by creating one large transaction with unlinkable inputs and outputs [30, 39]. As the mixing is only performed if all users agree on the transaction's correctness, this approach is much more secure than involving a trusted third party. However, single users can stall the mixing, which the other users must be able to detect to re-run the mixing without the

[1]Python-based implementation available at: *https://github.com/COMSYS/anonboot*

misbehaving user [39]. Another branch of cryptotumblers aims for providing a distributed mixing service [31, 54, 55], i.e., mix users' funds on their behalves without the risks involved with centralization. While Möbius [31] achieves this via an Ethereum smart contract, CoinParty [54, 55] implements a blockchain-external service via a shuffling network and secure multiparty computation (SMC), and thus can also be used for mixing cryptocurrencies without support for smart contracts, e.g., Bitcoin. In Figure 2c, we illustrate the operation of such a CoinParty-like distributed cryptotumbler. Using threshold signatures among the mixing peers prevents single adversaries from stealing funds, and secret-shared checksums are used to hold misbehaving mixing peers accountable during CoinParty's shuffling phase [55], e.g., if attempting to perform attacks known from shuffling networks (cf. Section 2.2). However, this additional protection can only tolerate adversaries controlling a share $f_S < 1/3$ of the service's privacy peers due to the application of SMC [4].

Takeaway. Although distributed cryptotumblers can increase the user's privacy, they either rely on smart contracts or are prone to Sybil attacks, i.e., single adversaries mimicking a distributed service. To the best of our knowledge, providing a technical medium to securely bootstrap cryptotumblers is still an open problem [39].

3 SCENARIO AND DESIGN GOALS

Based on existing anonymity services and the lack of proper bootstrapping processes, we now specify our scenario and design goals.

3.1 A Generalization of Anonymity Services

As we discussed in Section 2, technical means for securely bootstrapping distributed anonymity services are currently lacking. For a holistic solution resolving this lack of means to establish trust, we derive our scenario from the diverse landscape of existing services.

We assume a group of *privacy-aware users* who seek to utilize an anonymity service that increases their privacy on their behalf. To provide sufficient security and privacy guarantees, the users require that multiple independent operators of *privacy peers* jointly offer distributed anonymity services. Due to only limited scalability of network sizes of existing anonymity services, we assume that only a few privacy peers (e.g., < 100) provide services to much larger user groups. Service provision is thus prone to Sybil attacks.

To account for local user decisions, such as creating Tor circuits (cf. Section 2.1) or a minimum number of independent peers jointly providing a service, the user needs means to *securely discover* available peers and already established anonymity services. Furthermore, she has to *establish trust* in the faithful setup of those services even if she does now know the peer operators. Finally, the service discovery must allow for pooling users' anonymization efforts, as is required for shuffling networks or cryptotumblers. Additionally, we need to *incentivize* maintaining an honest majority of privacy peers. However, we assume that a share of privacy peers will still act maliciously and aim to, e.g., deanonymize users, stall the service, or inflict other damages such as theft through cryptotumblers.

In conclusion, users need to be ensured that they only utilize distributed anonymity services that act faithfully, i.e., the majority of the respective peers are honest. However, especially the setup and discovery of such services currently constitute weak points that adversaries could exploit to infiltrate anonymity services.

3.2 Design Goals for Secure Bootstrapping

The goal of our work is to create a decentralized medium for bootstrapping distributed anonymity services in a trustworthy manner and allowing privacy-aware users to discover both available peers and anonymity services. To achieve this goal, we identify the following main requirements and features.

(G1) ***Trustworthy Bootstrapping.*** Our medium must provide technical means to establish trust in available anonymity services and hence must be trustworthy itself. To this end, a decentralized design is reasonable to eliminate the need for users' trust in any dedicated medium operator. Furthermore, the medium must mitigate Sybil attacks to prevent its infiltration through adversaries. Finally, the medium must still remain in control over the setup of offered anonymity services through a secure bootstrapping procedure.

(G2) ***Secure and Lightweight Service Discovery.*** Our medium must only relay users to privacy peers and services that have been bootstrapped in a trustworthy manner. Previous approaches have proposed piggybacking node discovery for peer-to-peer systems onto a well-established decentralized medium such as IRC [21]. For such approaches, service discovery must limit its impact on the host system to facilitate the adoption of the bootstrapping process.

(G3) ***Broad Applicability.*** In Section 2, we discussed the variety of existing anonymity services. Consequently, we must account for this variety and allow users to discover and utilize different services for diverse applications. Finally, users should be able to use anonymity services corresponding to their individual preferences.

(G4) ***Scalability.*** Sufficiently large user bases are crucial to achieving high privacy levels via anonymity services. Our medium must thus effortlessly scale to large numbers of users and privacy peers.

(G5) ***Operator Incentives.*** Current honest anonymity services are typically offered on a voluntary basis [16]. However, if the effort of signaling honesty through our medium to publicly offer anonymity services becomes burdensome for operators, the number of volunteers might decrease. Hence, our medium must also consider the option to compensate for operators' efforts in its design.

4 ANONBOOT: A MEDIUM FOR SECURELY BOOTSTRAPPING ANONYMITY SERVICES

In this section, we first provide an overview of *AnonBoot* and then describe in detail how AnonBoot maintains a Sybil-resistant peer repository on top of a public host blockchain through standard transactions. Finally, we elaborate on how AnonBoot bootstraps anonymity services from this repository, i.e., how we elect privacy peers and then hand over control to the elected peers.

4.1 Design Overview

The main goal of AnonBoot is to provide a medium for *securely* bootstrapping distributed anonymity services that typically consist of only a few *privacy peers*. AnonBoot maintains a robust distributed *state* of available privacy peers and bootstrapping requests without storing privacy-compromising information. To this end, AnonBoot relies on the immutable ledger of a public *host blockchain* as the current state-of-the-art medium for communication and consensus without the need to rely on special trust in particular peers. By having privacy peers periodically *advertise* themselves on-chain through proof of work (PoW), AnonBoot maintains a *Sybil-resistant*

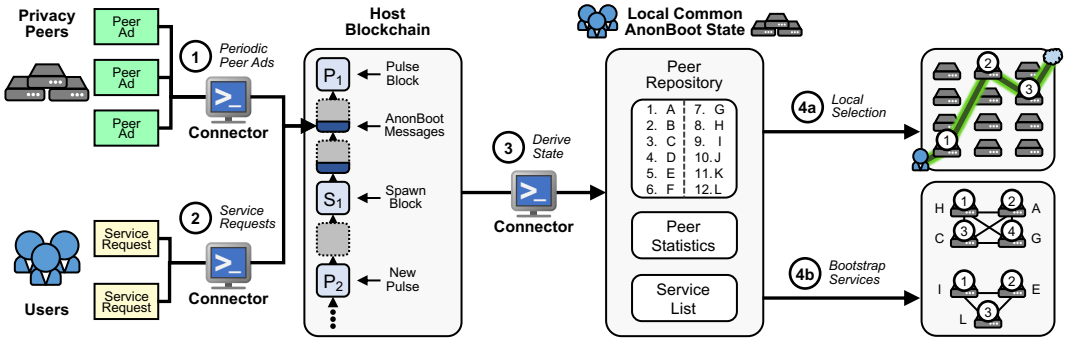


Figure 3: In AnonBoot, peers periodically advertise themselves on the host blockchain while solving small PoW puzzles to prevent Sybil attacks. Users can bootstrap different anonymity services, which are then utilized independently of AnonBoot.

peer repository. This way, the peer repository only contains recent privacy peers and adversaries need to invest resources in *maintaining* their influence rather than increasing it over time. AnonBoot realizes trustworthy bootstrapping **(G1)** by dynamically *electing* privacy peers based on these advertisements and further on-chain entropy. Thereby, AnonBoot prevents adversaries from manipulating peer election to gain an advantage over honest operators.

All participants locally operate a *connector* for all interactions with the host blockchain. The connector publishes new messages to the host blockchain and monitors it for new events. Based on these events, the connector updates AnonBoot's state. In this work, we detail how Bitcoin can be used as AnonBoot's host blockchain despite its very restricted intended ways to insert application-level data to show that AnonBoot can operate on top of most blockchains. Furthermore, the Bitcoin network is well-established with around 10 000 reachable nodes [52] vetting its blockchain and thus providing a strong trust anchor regarding AnonBoot's privacy peers.

In Figure 3, we present an overview of AnonBoot's bootstrapping process in four steps: First, in Step ①, privacy peers *advertise* themselves on the host blockchain. Subsequently, in Step ②, users *request* bootstrapping new anonymity services from a random set of advertised privacy peers. Next, in Step ③, all participants locally derive a common AnonBoot state. Finally, based on their state, users can either (Step ④a) *locally select* privacy peers for personal anonymity services without the need for full synchronization, or (Step ④b) privacy peers *bootstrap* a new shared anonymity service. We now provide a more detailed overview of these individual steps.

Periodic Peer Ads. In Step ①, AnonBoot creates a Sybil-resistant *peer repository* by requiring privacy peers interested in providing anonymity services to periodically issue *advertisements* on the host blockchain. Peer operators need to periodically refresh their advertisements at the start of each refreshment period, or *pulse*, while solving a small *PoW puzzle*. This core element of AnonBoot establishes a Sybil-resistant *peer repository* as peer operators need to invest their hardware resources at the start of each pulse to remain in the peer repository. To mitigate the advantage adversaries may gain through dedicated mining hardware, the exact design of the PoW puzzles is a crucial parameter of AnonBoot (cf. Section 6.1).

Service Requests. In Step ②, privacy-aware users may issue aggregatable on-chain *service requests* to request bootstrapping a shared anonymity service, e.g., a shuffling network or a crypto-tumbler, after a fixed-length negotiation phase. Service requests specify the type of the anonymity service as well as service-specific parameters such as minimum required sizes of anonymity sets.

Derive State. In Step ③, all participants locally process advertisements and service requests from the host blockchain to derive and verify AnonBoot's current state. By locally processing all on-chain service requests, all participants maintain a common *service list*. For processing requests, AnonBoot defines a randomized *peer election* that is inspired by blockchain sharding [22, 27, 53] to select random subsets of compatible privacy peers, which then jointly provide the requested service. Peer election is based on a pseudo-random number generator that is seeded with tamper-resistant entropy drawn from the host blockchain to enable all participants to locally derive the same service list. After discarding invalid or delayed messages, all participants obtain the same state, i.e., the current peer repository and statistics about previously discovered peers.

Local Selection & Service Bootstrapping. Step ④ finalizes the bootstrapping process provided by AnonBoot with two possible actions for users. Either, users directly perform an instant *local peer selection* only based on the peer repository (Step ④a), e.g., to establish a Tor circuit. Alternatively, users browse the service list (Step ④b) for securely bootstrapped anonymity services **(G2)**. Since all privacy peers derive the same state as users, they can check whether they were elected to provide a shared anonymity service and subsequently bootstrap these services by contacting other elected privacy peers. In both cases, communication is initiated through the AnonBoot connector, which then hands over the control entirely to the underlying anonymity protocol.

Our design ensures that AnonBoot only indexes anonymity services that are created in a Sybil-resistant manner as long as the peer repository itself consists of an honest majority. The maintenance of an honest peer repository is therefore essential to AnonBoot's security. To increase the willingness of honest peer providers to participate, AnonBoot can further decrease peer operators' costs by increasing the pulse duration, or existing anonymity services can be augmented with means for financial compensation, e.g., via anonymous micropayments [18]. We argue that the increased robustness against adversaries offered by AnonBoot is worthwhile for privacy-aware users even if they are required to compensate privacy peer operator's costs. However, AnonBoot is explicitly also operable by volunteers as long as its periodicity is tuned to prevent their recurring costs from becoming prohibitively high. In the following, we present AnonBoot's protocol design in more detail.

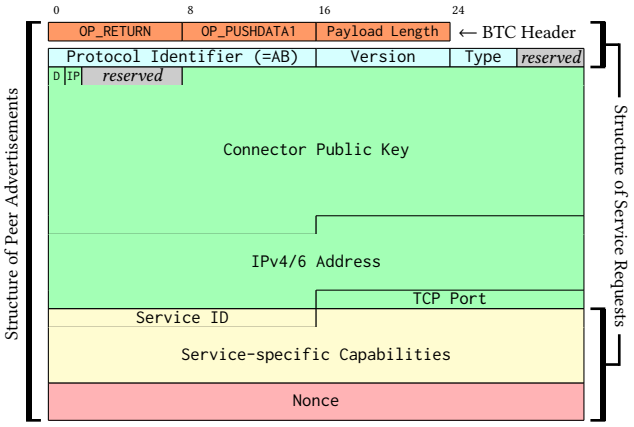


Figure 4: AnonBoot can run on top of Bitcoin using `OP_RETURN` transactions. Peer advertisements convey peers' contact information, capabilities, and the required PoW. Service requests bootstrap a service based on the capabilities and using the nonce as further entropy for peer election.

4.2 Sybil-Resistant Index of Peers and Services

AnonBoot relies on its host blockchain to maintain its Sybil-resistant peer repository and to instantiate new anonymity services based on users' requests. We now detail how AnonBoot can use Bitcoin as its host blockchain, only relying on standard transactions. All concepts carry over to other blockchains, especially to systems that can process arbitrary messages on-chain through smart contracts, e.g., Ethereum. After presenting the basic structure of AnonBoot's Bitcoin-compatible messages, we thoroughly describe the message layout of peer advertisements and service requests. Finally, we further elaborate on how AnonBoot enforces periodically refreshing messages to remain Sybil-resistant as well as fair toward honest privacy peers, and how it reduces its impact on the host blockchain.

Basic Message Layout. In Figure 4, we show the generalized structure of a Bitcoin-compatible AnonBoot message. Those messages are either *peer advertisements* or *service requests*. All messages consist of `OP_RETURN` Bitcoin transactions, which are allowed to carry up to 80 B of payload data [28]. This structure results in an unavoidable 3 B-long Bitcoin header consisting of the `OP_RETURN` operation and the payload's length [6]. The following AnonBoot header contains a protocol identifier (`AB`), as is common for `OP_RETURN`-based protocols [3], as well as the protocol version and message type. For extensibility reasons, we reserve four bits for future use.

Peer Advertisements. Privacy peers join AnonBoot's peer repository by periodically refreshing and publishing *peer advertisements* to the host blockchain. As we detailed in Figure 4, peer advertisements convey three main pieces of information for users and other privacy peers: (a) the peer's *contact information*, (b) its *capabilities*, and (c) a *solution* of its PoW puzzle. While sharing their capabilities and contact information is required for coordinating the peer election (Section 4.3), ensuring Sybil resistance via peer advertisements is crucial for AnonBoot's promised security properties.

First, each privacy peer announces the *contact information* of its connector so that users and other privacy peers can contact it securely in the following. The privacy peer announces its connector's public key as well as a pair of IP address and port for incoming connections. This indirection through a connector enables a unified connection interface for all anonymity services supported by AnonBoot. However, if the advertised service's required contact information fits into the peer advertisement, the privacy peer may set the `D`-flag to indicate the direct reachability of the service, i.e., the connector can be bypassed. By setting the `IP`-flag, the privacy peer toggles whether it is reachable via IPv4 or IPv6, respectively. Similarly, we reserved six additional bits for future use to remain flexible regarding other formats of contact information.

Second, each peer advertises its *capabilities*. These capabilities consist of a service identifier denoting which anonymity service the privacy peer supports as well as service-specific capabilities. This design supports the integration of a diverse landscape of anonymity services (cf. Section 2) as well as future services into AnonBoot and thus respects our requirement for broad applicability **(G3)**. These service-specific capabilities help users request services or locally select privacy peers that suit their individual needs. While smart contract-based host blockchains can process arbitrary messages and thereby enable the fine-grained expression of privacy peers' capabilities, the space limitations of Bitcoin's `OP_RETURN` payloads restrict this expressiveness. For instance, creating Tor circuits relies on potentially complex relay descriptors [43] that easily exceed available space and otherwise would impose a large overhead on the host blockchain. We make AnonBoot operable even in such restricted environments by allowing privacy peers to advertise coarse-grained capabilities as a browsing aid that is subsequently verified and refined via the participants' connectors.

Finally, privacy peers need to include a small PoW in their peer advertisements to *thwart Sybil attacks*. To be effective, the PoW must be cryptographically tied to the peer's identity as well as a recent point in the host blockchain to prevent an adversary from pre-computing or reusing peer advertisements. Only then, the PoW puzzle ensures that no peer can create disproportional numbers of peer advertisements compared to its hardware resources.

Service Requests. Users issue *service requests* to express that they want AnonBoot to bootstrap a new anonymity service corresponding to their requirements. Service requests closely resemble peer advertisements in their structure (cf. Figure 4), but they do not contain contact information. Further, the remaining fields are interpreted slightly differently. Through the capabilities, users express what service they intend to use as well as minimum requirements for the service to be bootstrapped. AnonBoot only allows users to request distinct classes of services through the capabilities to prevent a highly fragmented service list. In contrast to privacy peers, users do not solve a PoW puzzle in their service requests. Instead, users choose a random nonce, which AnonBoot will incorporate into its peer election to subsequently bootstrap the requested services. This way, users can further thwart attempts by adversaries to interfere with the peer election. A single service request will cause AnonBoot to instantiate the requested service to be used by an arbitrary number of users. Hence, AnonBoot easily scales to large user bases **(G4)**. However, users questioning the existing requests' randomness can issue redundant service requests and thus contribute to the entropy used for the peer election. AnonBoot aggregates redundant requests or similar requests superseded by service requests with stronger requirements, i.e., all requests' nonces influence the peer election,

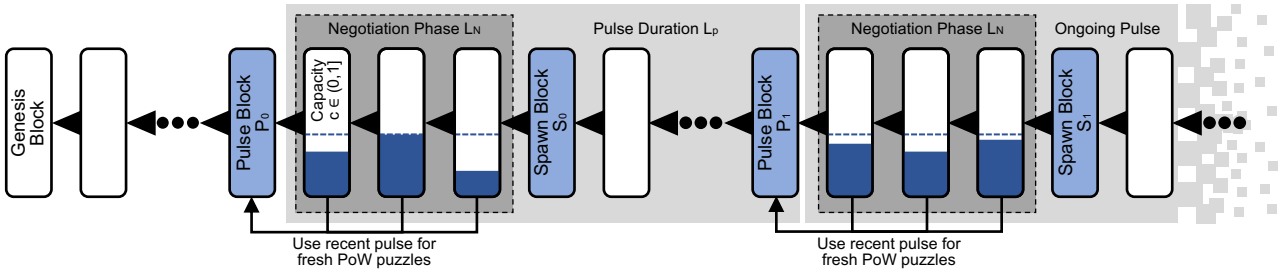


Figure 5: Peer advertisements are written to the host blockchain and must be renewed by the AnonBoot peers after each pulse, incorporating PoW over the most recent pulse block to ensure freshness. Only peer advertisements published during the negotiation phase are considered valid, where miners are advised to optionally not exceed the desired capacity of AnonBoot messages per block. New anonymity services are bootstrapped after the negotiation phase based on the next spawn block.

but only one service with the most restrictive capabilities of all aggregated service requests will be bootstrapped. At this point, we leave defining strategies to simultaneously instantiate multiple similar services as future work.

Pulse-based Message Release. In Figure 5, we illustrate AnonBoot's soft-state approach, which defines a *pulse* of length L_p in terms of block height on the host blockchain that triggers refreshing peer advertisements and accepts new service requests. Every p blocks, a new pulse starts and the most recent block on the blockchain serves as the new *pulse block*. Now all peers start to create their PoWs incorporating (a) their connector's public key, (b) a reference to the pulse block to ensure the freshness of advertisements, and (c) a nonce solving the PoW. To extract the maximum entropy from the pulse block, AnonBoot can apply extraction techniques, e.g., as proposed by Bonneau et al. [10].

For ideal fairness, all peers would have the same time window for providing a valid PoW. However, AnonBoot must be able to cope with a potential backlog of valid peer advertisements since we have no means to reliably enforce prioritized consideration of AnonBoot messages after a pulse block was being mined. Thus, we tolerate peer advertisements to be delayed throughout a *negotiation phase* of length L_N after each pulse. This length should be chosen as short as possible to prevent a devaluation of PoWs provided by honest peers, but it simultaneously should allow for including all anticipated peer advertisements in time even if single miners deliberately ignore AnonBoot messages. Furthermore, the negotiation phase provides some tolerance against accidental blockchain forks. While peers must recompute their PoW if the host blockchain discards the pulse block, a fork does not require AnonBoot to skip an entire pulse.

The tunable duration of each pulse with its associated negotiation phase also allows to adjust the burden put on its participants as well as its host blockchain in a fine-grained manner and thus allows keeping the service discovery lightweight **(G2)**. First, AnonBoot disincentivizes excessive creation of messages as honest peers will ignore all messages outside of a pulse's negotiation phase. Second, increasing L_p without changing L_N reduces the number of messages required to maintain the peer repository, i.e., costs for all peers are reduced, without weakening AnonBoot's Sybil resistance and only at the cost of the peer repository becoming less flexible. However, AnonBoot still releases messages in bursts at the start of each pulse. If these occasional message bursts prove to be burdening the host blockchain, AnonBoot-aware miners can follow an optional guideline to accept messages only up to a per-block *capacity* $c \in (0, 1]$ without impacting AnonBoot negatively. Furthermore, more awareness from miners on the host blockchain has the potential to further reduce costs of AnonBoot peers and thereby lower the bar for altruistic peer operators. Either through updated consensus rules or novel, AnonBoot-tailored blockchain designs, miners can be incentivized to reserve up to $c \cdot 100\%$ of their blocks during each negotiation phase for including AnonBoot messages at no costs. For instance, full nodes may then reject blocks that ignore a current backlog of pending AnonBoot messages. We further quantify how the host blockchain can steer the impact of AnonBoot in Section 7.2. While this approach requires that miners are not entirely oblivious of AnonBoot, it ensures that AnonBoot can operate at minimal costs without burdening the host blockchain.

4.3 Bootstrapping Secure Anonymity Services

All privacy peers that regularly refresh their peer advertisements are eligible for providing anonymity services. In this section, we describe how AnonBoot facilitates bootstrapping anonymity services based on the current pulse and its resulting peer repository. After briefly describing how control is handed over from AnonBoot to its bootstrapped services, we consider users locally picking privacy peers directly from the peer repository and then provide details on how AnonBoot elects privacy peers to bootstrap publicly available, distributed anonymity services.

Bootstrapping Users and Privacy Peers. AnonBoot provides only a medium for establishing and finding trustworthy distributed anonymity services. Its responsibility thus also involves enabling users to contact privacy peers that provide the requested anonymity service. In most cases, peer advertisements will announce the contact information of the involved privacy peers' AnonBoot connector. During the handover of control via her own connector, the user verifies the correctness of each peer's contact information, especially whether it possesses the private key corresponding to its advertisement. If successful, the connectors perform a service-specific handover so that further interaction is now performed entirely according to the anonymity service protocol. In cases where indirection through the connector is undesired, privacy peers may use the `D`-flag (cf. Section 4.2) to signal that the contact information directly corresponds to the endpoint of its offered service. However, a Bitcoin-backed AnonBoot only supports `OP_RETURN`-based direct advertisements if they can hold all required contact information.

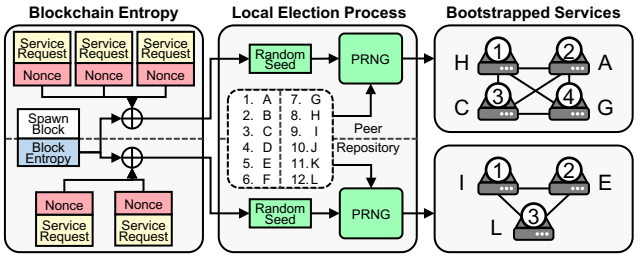


Figure 6: A pseudo-random peer election based on blockchain entropy enables all participants to locally compute the same service lists from the peer repository.

Depending on the particular anonymity service (cf. Section 2), users either contact (a) only one privacy peer, (b) all privacy peers of one anonymity service, or (c) may only indirectly contact subsequent privacy peers for security reasons, e.g., when establishing Tor circuits. In cases where a direct connection to peers is prohibited, users can interleave bootstrapping with the anonymity service and incrementally contacting the new peers' connectors. For instance, Tor builds circuits hop by hop [46], and thus users can contact the connectors of subsequent Tor nodes via partially established circuits, which aligns well with Tor's design [46].

Local Selection of Peers. The peer repository's Sybil resistance (cf. Section 4.2) makes it a suitable replacement for centrally maintained directories. Privacy-aware users individually monitor peer advertisements, which enable them to instantly select privacy peers based on their local view on the peer repository, i.e., this peer selection is independent of AnonBoot's pulses. Furthermore, users may base their decisions on individual security and privacy preferences, e.g., they only select privacy peers who recently advertised themselves, or they may locally keep track of peer statistics, such as their first occurrence or how regularly they refresh advertisements.

When selecting privacy peers, the user verifies the correctness of those peers' advertisements and contacts their connectors. To this end, users only have to passively monitor the host blockchain for valid peer advertisements from the current pulse. Each peer that (a) performed a valid and fresh PoW, (b) is reachable via its connector's contact information, and (c) advertised a valid corresponding public key is eligible to be selected by the user. Ultimately, the user randomly selects a sample of peers she considers eligible replacing any inaccessible peers until the service can be provided correctly.

Service Requests For Peer Election. AnonBoot derives the demand for anonymity services from users' service requests during the negotiation phase (cf. Section 4.2). Based on these service requests, we must ensure that peers are chosen randomly in a transparent manner to provide a secure bootstrapping process. We achieve this requirement through a locally replicable *peer election* process that relies on a pseudo-random number generator (PRNG) and seeds derived from random values on the host blockchain. This way, all participants obtain the same list of elected peers for each distinct service request for subsequent coordination.

In Figure 6, we present AnonBoot's peer election in more detail. To derive the seed, we rely on two sources of entropy. On the one hand, users submit 8 B-long nonces with their service requests. We aggregate the nonces of all matching service requests during one pulse, i.e., requests for the same anonymity service with the compatible capabilities. This approach allows a bootstrapping of anonymity services with a single service request for efficiency while it also offers privacy-aware users the chance to directly influence the peer election's randomness without spawning concurrent services that are potentially under-utilized. On the other hand, we consider the *spawn block* of each pulse, i.e., the first block *after* the pulse's negotiation phase has concluded. Thus, an adversary cannot craft nonces to bias the peer election without mining the spawn block. We incorporate entropy from this block into the seed for the PRNG to ensure its freshness. All participants locally use the PRNG with this seed to elect peers for each service request and select a pseudo-random sample of privacy peers from the peer repository that is compatible with the service request. A common ordering of the peer advertisements ensures that all participants select the same samples. The peer election allows all participants to compute the same service list and thus synchronize in a decentralized manner. Hence, AnonBoot helps users find the required entry points for using anonymity services, and the privacy peers learn whom to connect to when being elected to join a specific anonymity service.

Conclusion of Design. Our design of AnonBoot enables trustworthy bootstrapping **(G1)** since (a) it operates on top of a public host blockchain in a decentralized manner, (b) it mitigates Sybil attacks through periodically refreshed and PoW-based peer advertisements, and (c) it realizes a secure bootstrapping process using entropy from users as well as the host blockchain's mining process. By exchanging messages through the host blockchain, our design facilitates secure service discovery with only a low impact on the host blockchain due to AnonBoot's parametrizable pulse length and per-block capacity **(G2)**. Our protocol-agnostic message structure and handover of control moreover ensure a broad applicability of AnonBoot **(G3)**. Finally, AnonBoot scales to large user bases as single service requests suffice to bootstrap anonymity services usable by arbitrarily many users **(G4)**. In the following, we outline how to integrate different anonymity services into our medium and how AnonBoot can incentivize honest participation of privacy peers to satisfy the remaining design goal **(G5)**.

5 REALIZING USE CASES IN ANONBOOT

After presenting the general medium provided by AnonBoot, we now discuss how the established anonymity services, which we presented in Section 2, can operate on top of this medium regarding the achievable benefits, the technical integration, and how to financially incentivize honest privacy peers' participation **(G5)**. First, we discuss how AnonBoot's peer repository can realize a distributed directory service for anonymity networks. Subsequently, we discuss the bootstrapping of shuffling networks and cryptotumblers, as both behave similarly in AnonBoot.

5.1 Decentralized Onion Routing via AnonBoot

AnonBoot's Sybil-resistant peer repository constitutes a cryptographically controlled replacement for otherwise logically centralized directory services. Hence, our approach is beneficial if users expect operators to be corruptible or malicious. Nevertheless, AnonBoot must allow users to still make informed choices about the establishment of circuits, and we must account for the infeasibility and insecurity of users directly contacting all peers of a circuit.

Benefits. In currently deployed anonymity networks, the directory service is essentially centralized. For example, Tor is pre-shipped with a hard-coded list of currently ten *directory authorities* [44], which jointly maintain its directory [43]. This approach leaves current anonymity networks vulnerable to viable attacks on the directory service [16]. Contrarily, AnonBoot allows creating a fully decentralized directory that is implicitly maintained through the host blockchain and locally verifiable by all AnonBoot participants. Based on this directory, users can locally select privacy peers for their circuits as they currently do through Tor's directory service.

Peer Advertisements. Privacy peers can advertise themselves as onion routers. However, Tor's directory service maintains extensive meta information about available peers [43], which in most cases cannot be encoded in a single `OP_RETURN`-based peer advertisement as required when AnonBoot shall operate on top of Bitcoin. Among this meta information is the peer's contact information, cryptographic identity, available bandwidth, supported features, and exit policies, i.e., access control list for connections to hosts on the public Internet [43]. We thus make use of the peer advertisements' capabilities (cf. Section 4.2) to encode an *overview* of the peers' full meta information. This overview is a coarse summary of a privacy peer's *advertised* capabilities and should be indicative of its actual capabilities. Users can then browse available privacy peers based on these advertised capabilities without additional delays. When establishing a new circuit, the user should then request the chosen privacy peers' full server descriptors, verify that this descriptor matches the previously advertised capabilities, and check that the full descriptor is also compatible with the user's requirements.

Bootstrapping Phase. The circuits users establish within anonymity networks are intended to provide sender-receiver anonymity. Hence, a critical constraint is that users only communicate directly with the first peer of a circuit. AnonBoot naturally integrates with the resulting incremental circuit establishment of Tor [46]: The user incrementally establishes the next hop of her new circuit based on her selected peers' advertisements. She contacts the new peer through her partially established circuit and attempts to hand over control to the Tor client through the connector. If this handover of control fails, e.g., due to an invalid advertisement, she terminates the connection to that peer and selects a replacement privacy peer. Although an honest majority among privacy peers reduces the overhead of such security back-offs, enabling privacy peers to build up a positive reputation across consecutive peer advertisements promises to further reduce respective risks for users.

Incentives. If honest providers of onion routers must be compensated for investing their resources to periodically solve PoW puzzles and advertise themselves in AnonBoot, cryptocurrency-based service fees are a promising means for creating operator incentives. However, on-chain payments bear high risks of implicitly recording information about users' circuits irrevocably. We thus propose that users and privacy peers create anonymous unidirectional micropayment channels [18]. Although micropayment channels require an on-chain setup, users can protect their privacy due to the concurrent setup transactions of all users. This way, users can pay peers who advertise themselves via AnonBoot for their service.

5.2 Shuffling Networks and Cryptotumblers

AnonBoot's main advantage is to provide a medium for bootstrapping distributed anonymity services and to ensure their privacy peers' independence through its PoW puzzles and secure peer election. Privacy-aware users thus gain the opportunity to rely on secure on-demand anonymization for, e.g., message shuffling or increasing their financial privacy.

Benefits. Distributed systems that outsource responsibility to a set of peers typically rely on secure multi-party computation (SMC) [1, 55]. Unfortunately, scalability limitations of those SMC protocols hinder distributing responsibility among large sets of privacy peers. Without carefully selecting the responsible privacy peers, insider adversaries thus can gain power and cause harm relatively easily. However, our considered use cases of anonymous message disclosure and tumbling cryptocurrencies lack a trustworthy peer selection process, and adversaries are highly incentivized to attack such systems. For example, an adversary could easily spawn numerous interconnected privacy peers, and thereby mimic a distributed cryptotumbler, tricking users into participation. AnonBoot provides the ingredients to *cryptographically ensure* through its Sybil-resistant peer repository and locally verifiable peer election that an adversary cannot bootstrap malicious services. Hence, privacy-aware users reduce their individual risks when utilizing distributed anonymity services bootstrapped via AnonBoot.

Peer Advertisements. The capabilities privacy peers need to advertise highly depend on the provided anonymity service. Similarly to our previous use case, privacy peers should facilitate the users' browsability of anonymity services by advertising supported policies or security parameters. However, AnonBoot does not consider the service-specific capabilities during its peer election but requires the service identifier used (cf. Section 4.2) to ensure compatibility among privacy peers advertising the same service.

Bootstrapping Phase. Privacy peers are partitioned by the identifier of the anonymity service they advertise to ensure compatibility during the bootstrapping process. By locally replaying the peer election, each privacy peer gets to know (a) whether it was elected to provide a service, (b) which peers are elected to bootstrap the same service instance, and (c) the peer's logical position within the new network. Hence, privacy peers can independently configure and bootstrap the anonymity service. Currently, we take a conservative approach and declare services stale after a couple of pulses to mitigate the impact of privacy peer churn and malicious services bootstrapped by chance. However, conceptually, AnonBoot also supports bootstrapping long-lived anonymity services.

Incentives. Since these use cases do not prohibit a direct connection between users and elected privacy peers, we can simplify our payment scheme proposed in Section 5.1 and instead require users to pay an upfront fee (e.g., as proposed by CoinParty [55]). We argue that the increased security provided by AnonBoot is worth compensating the privacy peer's efforts of solving PoW puzzles.

Takeaway. In conclusion, AnonBoot provides a viable medium for bootstrapping anonymity services from a diverse set of available applications as it simultaneously mitigates malicious influences and compensates honest operators if privacy peers.

6 SECURITY DISCUSSION

We assess AnonBoot's robustness against adversaries by discussing the implications of incorporating PoW into peer advertisements and arguing that active adversaries cannot bias the peer election.

6.1 Proof of Work Against Sybil Attacks

Requiring a PoW in each peer advertisement hampers an adversary's effort to control large portions of the peer repository and thus his overall influence. However, the choice of the PoW scheme is paramount for AnonBoot's resilience against Sybil attacks. We thus highlight the need for an appropriate PoW scheme but leave its final instantiation to be adapted to users' needs in future work.

Particularly, AnonBoot's PoW scheme must ensure that operators can only create peer advertisements at rates corresponding to their number of physical devices controlled while not excluding honest operators using commodity hardware. While specialized hardware is known to provide huge advantages for CPU-bound PoW schemes such as Bitcoin's scheme, memory-bound PoW schemes such as Ethereum's Ethash [42, 51], Cuckoo Cycle [47], Equihash [5], or RandomX [41], which was recently adopted by Monero [40], are promising candidates to be adapted for utilization with AnonBoot. For instance, based on `openssl speed`, we observe that a server (two Intel Xeon Silver 4116, 187.39 GiB RAM) outperforms a commodity desktop PC (Intel Core 2 Q9400 CPU, 7.67 GiB RAM) by two orders of magnitude for Bitcoin's HASH256-based PoW scheme. Further, Bitcoin mining hardware [7] reportedly outperforms our commodity PC by eight orders of magnitude, which clearly underlines the potential advantage of adversaries relying on specialized hardware to forge advertisements using CPU-bound PoW schemes.

Contrarily, initial measurements using Ethash (via `geth`'s CPU-based mining) and RandomX indicate that the same server only achieves a mere 7.5× (12.7×) speed-up over the desktop PC in terms of achievable hash rate using this PoW scheme. Thus, relying on memory-hard PoW schemes is preferable to prevent adversaries with powerful devices or, e.g., a botnet, from increasing their influence on the peer repository in an incommensurate manner [5].

Finally, we address the challenge of steering the PoW puzzles' difficulty to account for improvements in hardware capabilities. In contrast to cryptocurrency mining, AnonBoot's peer advertisements have no inherent concurrency, i.e., the size of the peer repository does not influence the required difficulty for the PoW. Assuming an honest majority, we can expect that privacy peers have an interest in keeping an appropriate PoW difficulty for security reasons. Thus, we can dedicate unused bits in the peer advertisements (cf. Section 4.2) to enable voting on increasing the difficulty. Privacy peers would then update their local threshold for accepting the PoW in peer advertisements based on votes of the (honest) majority.

Takeaway. Utilizing a simple CPU-bound PoW scheme for our puzzles would significantly impact AnonBoot's security properties. Contrarily, memory-bound PoW schemes constitute a secure building block to maintain a Sybil-resistant peer repository. As for existing systems, such as Tor or Bitcoin, the reliability of AnonBoot's peer repository then depends on maintaining an honest majority, either on a voluntary basis or through operator incentives. Finally, we can further leverage this honest majority to implement a self-regulated adaption of the puzzles' difficulty.

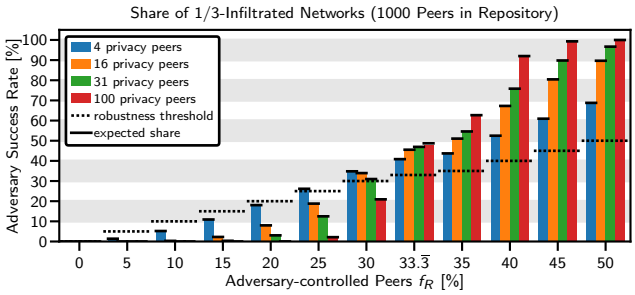


Figure 7: The success rate of an adversary to 1/3-infiltrate services by chance can be kept below the robustness threshold $T_{1/3}$ for $f_R \leq 30\%$, i.e., peer election remains robust for relevant scenarios involving SMC-based anonymity services.

6.2 Security of Bootstrapped Services

The core design goal of AnonBoot is to securely bootstrap distributed anonymity services. We have already shown that AnonBoot can maintain a Sybil-resistant peer repository, i.e., adversaries cannot control a disproportional fraction of the peer repository. However, adversaries can still enter the peer repository as long as they can create a valid PoW. We now highlight that AnonBoot's peer election is robust against adversarial bias and that bootstrapped anonymity services can tolerate a share of adversarial privacy peers.

Security of Local Peer Selection. Anonymity services which rely on local peer selection only require the Sybil resistance provided by AnonBoot's peer repository (cf. Section 6.1). However, privacy peers are not always treated equally: For example, in Tor, users change their first relays, i.e., guard nodes, only infrequently [26] and they can only use exit nodes that support their requests [16]. By encoding the privacy peers' capabilities accordingly (cf. Section 4.2), users can respect these properties when establishing circuits. The peer repository hence constitutes a secure alternative to current directory services provided by trusted third parties.

Robustness of Peer Election. AnonBoot's peer election must properly protect its users, i.e., bootstrap secure anonymity services. To assess the influence of an adversary, we consider his chance of *infiltrating* an anonymity service during peer election based on the share of privacy peers he controls. An adversary successfully infiltrates an anonymity service if he controls a share $f_S \geq t_I$ of that service's privacy peers, exceeding its *infiltration threshold* t_I, i.e., he can defy the service's underlying security guarantees. E.g., a malicious adversary infiltrates any SMC-based anonymity service when controlling $f_S \geq 1/3$ of the peers [4]. Under this notation, we consider peer election to be robust if adversaries cannot increase their chance of infiltrating services beyond their share f_R of privacy peers in the peer repository. More formally, assuming that no adversarial share of the peer repository exceeds a threshold t_R, we define a *robustness measure* $\mathcal{R}(t_I, t_R) = 1\text{–}\Pr(f_S \geq t_I \mid f_R \leq t_R)$. We further define that the peer election is *robust* iff $\Pr(f_S \geq t_I \mid f_R \leq t_R) \leq t_R =: T_I$ holds, i.e., $T_I = t_R$ can be interpreted as a *robustness threshold* against t_I-infiltration. For instance, an adversary controlling up to $f_R \leq 10\%$ of the peer repository should only have a chance of $T_I = 10\%$ to t_I-infiltrate an anonymity service.

Figure 7 highlights AnonBoot's robustness regarding SMC-based anonymity services, i.e., $t_I = 1/3$ [4], which can tolerate up to

$\lfloor n/3 \rfloor$ adversary-controlled privacy peers. For desired networks consisting of 4, 16, 31, and 100 peers (i.e., $t_I = 1, 5, 10, 33$) respectively, we measured the success of an adversary controlling a growing share f_R of the peer repository to infiltrate anonymity services by chance due to our peer election. To extract entropy from the pulse's spawn blocks, we rely on the Merkle tree root. More secure entropy extraction can be achieved by applying more sophisticated randomness extractors [10]. For our evaluation, we assume a peer repository consisting of 1000 peers, randomly elect peers for 100 000 anonymity services for each scenario, and count the number of 1/3-infiltrated services. We also highlight the robustness threshold for comparison and provide the expected shares of 1/3-infiltrated services based on combinatoric considerations.

Our evaluation reveals two major findings: First, our peer election is *fair* in that it almost perfectly yields the expected distribution of 1/3-infiltrated services when electing honest and dishonest peers uniformly at random. Second, the peer election remains robust as long as the adversary controls $f_R \leq 25\%$ of the peer repository. For a growing power of the adversary, AnonBoot cannot guarantee robustness, although larger anonymity services yield better protection if the adversary controls a share of at most $f_R \leq 30\%$. For all $f_R \geq 1/3$, AnonBoot is not robust anymore as the adversary can infiltrate most SMC-based services. However, in those cases, his control of the peer repository exceeds the infiltration threshold for SMC-based services; thus, we consider the peer repository insecure.

AnonBoot relies on entropy from the host blockchain to seed its PRNG for peer election. Adversaries are thus tempted to influence the seed by interfering with the on-chain data to increase their chances of infiltrating anonymity services. Our rationale for AnonBoot's robustness only holds if we can effectively prevent such interference. As we described in Section 4.3, we include user-submitted entropy into the seed derivation to ensure that seeds are not entirely determined by the miners of AnonBoot's host blockchain. However, by incorporating the spawn block, we, in return, drastically limit the capabilities of an adversary. Namely, the adversary must (a) successfully mine the spawn block S_i for pulse P_i, while (b) crafting this block to yield, in conjunction with the user-supplied entropy, a biased pre-image of a favorable seed, which is (c) derived from a cryptographic hash function. Assuming that no adversary possesses the computing power to control the host blockchain, we deem this kind of attack economically infeasible as honest mining is more profitable for the adversary. In the future, we could also adapt AnonBoot to consider multiple consecutive spawn blocks to further thwart the influence of adversaries.

Security of Handover Process. For most anonymity services, AnonBoot requires indirection through the connector when first establishing connections. During this handover process, each participant's connector has to authenticate all privacy peers based on the public key previously announced in the respective peer advertisements. Hence, users only connect to privacy peers controlled by operators that created valid and distinct peer advertisement. The adversary thus cannot launch Sybil attacks through this indirection.

Denial of Service (DoS). Due to our secure local peer selection, robust peer election, and secure handover primitives, the security of utilizing bootstrapped anonymity services only depends on the security guarantees offered by those services. While AnonBoot prevents adversaries from infiltrating anonymity services with high probability, distributed services are still prone to DoS attacks, effectively preventing proper anonymization. However, we argue that the anonymity services currently covered by AnonBoot can cope with such attacks: First, AnonBoot allows for the efficient creation of circuits for anonymity networks. Hence, the limited influence of single stalling relays does not significantly impede the users' privacy. Second, CoinParty, our investigated cryptotumbler, detects and excludes stalling peers as long as adversaries did not infiltrate at least 1/3 of the peers of the CoinParty instance's mixing network [55]. Finally, while traditional shuffling networks do not provide protection against DoS attacks, extending them with the measures taken by CoinParty achieves the same level of protection. Thus, our peer election does not directly thwart DoS attacks, but their impact on our considered anonymity services is highly limited.

Takeaway. In conclusion, the peer election yields trustworthy anonymity services as long as the majority of eligible privacy peers contribute honestly to providing these services, which we ensure through our Sybil-resistant peer repository and operator incentives.

7 PERFORMANCE EVALUATION

We demonstrate AnonBoot's feasibility by discussing its required synchronization times and its impact on its host blockchain.

7.1 Time Overheads

To continually monitor AnonBoot's state, participants should maintain a local copy of its host blockchain. However, we only rely on the correctness of the host blockchain's PoW as AnonBoot's trust anchor. Hence, while constrained devices may rely on a trusted source to provide a correct state, e.g., a trusted IoT gateway [20], more powerful devices preferably maintain their AnonBoot state themselves. We again consider Bitcoin as our working example for a host blockchain and highlight how initial synchronization with AnonBoot differs from a full synchronization with Bitcoin. Since the validity of peer advertisements typically expires in AnonBoot, as with block-pruning approaches [29] participants only have to download and verify the chain of Bitcoin's block *headers* and process only a few recent full blocks to derive their state by searching for AnonBoot-related `OP_RETURN` transactions. The required number of blocks to process depends on the pulse length L_p, the validity period of single peer advertisements, and the maximum lifespan of bootstrapped anonymity networks. Even if bootstrapped services remain active indefinitely (cf. Section 5.2), new users can still start synchronizing from only recent blocks and afterward discover older services from the remaining blocks in the background. After this initial synchronization, participants actively monitor the host blockchain for new AnonBoot messages. This overhead is negligible for Bitcoin as new blocks are only mined every ten minutes [48].

This potentially slow block creation interval, however, introduces unavoidable delays for the bootstrapping of new anonymity services as services are only created once a pulse's spawn block has been mined (cf. Figure 5). For instance, a pulse length of $L_p = 12$ blocks and a negotiation phase of $L_N = 3$ blocks on top of Bitcoin means that privacy peers have at most 30 min to solve their PoW puzzle, but in the worst case users have to wait up to 2 h until their requested anonymity service starts bootstrapping. Regardless, our relevant use cases of shuffling networks and cryptotumblers are

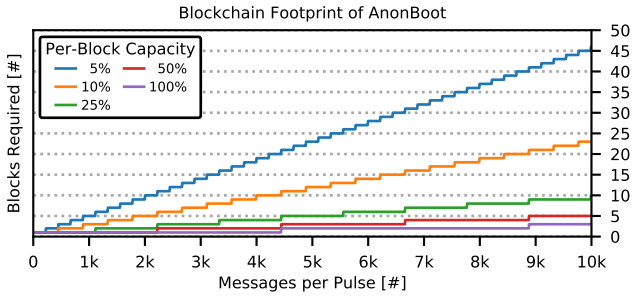


Figure 8: AnonBoot scales to thousands of messages per pulse with only small impact on Bitcoin as its exemplary host blockchain even for constrained per-block capacities.

latency-tolerant and sometimes even deliberately stretch their operation over time to further increase the level of achieved privacy [55]. If more timely service utilization is required, users can consider services valid for longer periods, thereby reducing the impact of the inflicted one-time overhead. In this case, users have to trade off delays against security as longer validity periods devalue the protection offered by periodic PoW puzzles.

Contrarily, local peer selection only depends on individual user decisions and thus only relies on knowing a recent valid state of the peer repository, i.e., full synchronization up to the most recent pulse is desirable but not required. Namely, users can instantly sample privacy peers based on their current state, and thus AnonBoot preserves the low-latency requirement of anonymity networks.

Takeaway. We conclude that AnonBoot (a) has low synchronization overhead, (b) introduces feasible latencies for bootstrapping anonymity services, and (c) supports instant local peer selection.

7.2 Small Blockchain Footprint and Low Costs

We now show that AnonBoot realizes lightweight service discovery **(G2)** by assessing its impact on the host blockchain. We measure the blockchain footprint of AnonBoot using Bitcoin's regression test mode. As we discussed in Section 4.2, the per-block capacity c helps to trade off how much transaction bandwidth AnonBoot consumes and the duration L_N of the negotiation phase.

In Figure 8, we illustrate how the minimal required L_N grows depending on the number of used AnonBoot messages and the capacity c. On average, an `OP_RETURN` transaction for one peer advertisement or user request has a size of 307 B and a weight of 901 WU (weight units). Since the introduction of Segregated Witnesses, the notation of block weight (limit 4 million WU) superseded the old measure of the block size (limit 1 MB) [8]. For our measurements, we fill Bitcoin blocks while respecting the allowed capacity c. Our results reveal that AnonBoot easily scales to large peer repositories and user bases with only a small footprint on Bitcoin. When using a per-block capacity of only 10 %, AnonBoot can support up to 10 000 messages during a negotiation phase with $L_N = 23$.

Peer repositories of size 1000, which are already sufficiently secure as we demonstrated in Section 6.2, have an only negligible impact on Bitcoin: For a small capacity of only 5 % to account for Bitcoin's low overall transaction throughput the negotiation phase still concludes after $L_N = 5$ blocks with space for up to 109 user requests. We expect only a few user requests as a single request suffices to bootstrap a service. The scalability then only depends on the upper-layer protocol used and is independent of AnonBoot.

Finally, we briefly consider the costs inflicted by fees when privacy peers and users publish their `OP_RETURN` transactions to leverage Bitcoin's consensus properties. Albeit fluctuating, the current (March 8, 2020) recommended fee is 6 satoshi per byte (1 satoshi $= 10^{-8}$ BTC) [17], and Bitcoin's market price is around 9067 USD [9]. Hence, a peer advertisement currently costs an operator 0.17 USD, which AnonBoot can amortize through larger pulse lengths while keeping the negotiation phase, e.g., of multiple days.

Takeaway. Overall, our analysis shows that AnonBoot can bootstrap over 100 services from a peer repository of size 1000, serving potentially thousands of users, and can scale well beyond this size with only a small impact on Bitcoin as its host blockchain.

8 RELATED WORK

The bootstrapping problem and Sybil attacks are inherent for distributed protocols. In 2007, Knoll et al. [21] surveyed different approaches to finding entry points for established peer-to-peer networks. Among other approaches, the authors proposed to bootstrap nodes through a distributed host system such as IRC [21]. Orthogonally, Levine et al. [25] reviewed approaches to mitigate Sybil attacks. From this taxonomy, only resource testing and recurring costs and fees are applicable to fully decentralized systems without further assumptions. Recurring costs, namely periodic PoW-based refreshments of eligibility, are a familiar building block in the field of blockchain sharding [22, 27, 53], where responsibilities to verify the proposed transactions are distributed among full nodes over time to improve scalability. AnonBoot adapts this Sybil-resistant building block in the form of peer advertisements to implement the novel application that is securely bootstrapping distributed anonymity services. In line with Knoll et al. [21], we publish these periodic advertisements through a public blockchain as AnonBoot's host system and trust anchor. This choice allows us to massively reduce the coordination complexity in AnonBoot since blockchains already offer a distributed means to reach a consensus of state.

Recently, Lee et al. [24] proposed that the user's ISP could provide privacy services, such as address hiding or VPN tunneling. This work is orthogonal to our approach as we bootstrap services without relying on a dedicated central operator. Namely, AnonBoot can also help users to increase their privacy against the ISP itself.

As one of its applications, AnonBoot realizes a decentralized directory service for anonymity networks such as Tor. Similar contributions were made by other works, e.g., NISAN [36] or ShadowWalker [33]. However, while both proposals prevent adversarial bias, they do not feature AnonBoot's protection against Sybil attacks. Furthermore, these approaches do not address the challenges of heterogeneous privacy peers, such as Tor nodes with different exit policies. AnonBoot introduces the capabilities in its peer advertisements specifically to overcome this shortcoming. While approaches to realize sticky data policies on how to handle user privacy [37] are related to our specification of peer capabilities, even highly compressed policies such as provided by CPPL [19] may exceed our space limitations, especially when relying on Bitcoin's `OP_RETURN` transactions to operate AnonBoot. Although CPPL may facilitate simple peer capabilities, more complex instances, such as Tor relay descriptors, require manual capability abstractions.

9 CONCLUSION

We introduced AnonBoot, a blockchain-based medium to securely bootstrap distributed anonymity services via already established public blockchains, such as Bitcoin, as a trust anchor. All AnonBoot peers communicate with each other through on-chain transactions, and, thereby, they are able to derive the same local view on AnonBoot's state by simply monitoring the host blockchain. Our design allows for discovering peers to create Tor circuits as well as to bootstrap shuffling networks and distributed cryptocurrency tumblers on demand. AnonBoot achieves its resilience against adversaries by two core mechanics: First, Sybil attacks are thwarted by forcing peers to periodically refresh their membership in a repository of peers who are eligible to provide anonymity services while including a memory-bound, and thus fair, proof of work. Second, an adversary who joins this peer repository cannot bias the peer election for new anonymity services since this peer election is based on user inputs as well as future blocks from the host blockchain.

The evaluation of our Bitcoin-based prototypic implementation of AnonBoot shows that public blockchains constitute a well-suited foundation for bootstrapping distributed systems: AnonBoot can easily maintain a peer repository consisting of 1000 peers on top of Bitcoin, managing services for potentially thousands of users. These results show that AnonBoot can operate on top of most blockchains, even if they have limited capabilities to store application-level data.

In the future, AnonBoot's utility can be further increased by identifying novel use cases apart from anonymity services. AnonBoot lends itself to bootstrapping any distributed service, e.g., to distribute trust in other domains via secure multiparty computation.

ACKNOWLEDGMENTS

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under funding reference numbers 16DHLQ013 and Z31 BMBF Digital Campus. The funding under reference number Z31 BMBF Digital Campus has been provided by the German Academic Exchange Service (DAAD). The responsibility for the content of this publication lies with the authors. The authors thank Jöran Wiechert for his support with the prototype.

REFERENCES

[1] Giuseppe Ateniese et al. 2017. Redactable Blockchain – or – Rewriting History in Bitcoin and Friends. In *Proc. IEEE EuroS&P*.
[2] Badbitcoin.org. 2014. *The Definitive Bitcoin and Cryptocurrency Fraud List.* Accessed: 03/10/2020.
[3] Massimo Bartoletti and Livio Pompianu. 2017. An analysis of Bitcoin OP_RETURN metadata. In *Proc. IFCA FC Bitcoin Workshop*.
[4] Michael Ben-Or et al. 1988. Completeness Theorems for Non-cryptographic Fault-tolerant Distributed Computation. In *Proc. ACM STOC*.
[5] Alex Biryukov and Dmitry Khovratovich. 2017. Equihash: Asymmetric Proof-of-Work Based on the Generalized Birthday Problem. *Ledger* 2.
[6] Bitcoin Wiki. 2010. *Script.* bitcoin.it, accessed 03/10/2020.
[7] Bitcoin Wiki. 2015. *Mining Hardware Comparison.* bitcoin.it, accessed: 03/10/2020.
[8] Bitcoin Wiki. 2017. *Segregated Witness.* bitcoin.it, accessed: 03/10/2020.
[9] Blockchain.com. 2011. *BTC to USD: Bitcoin to US Dollar Market Price.* blockchain.com/charts/market-price, accessed 03/10/2020.
[10] Joseph Bonneau et al. 2015. On Bitcoin as a public randomness source. *IACR Cryptology ePrint Archive* 2015.
[11] Joseph Bonneau et al. 2014. Mixcoin: Anonymity for Bitcoin with Accountable Mixes. In *Proc. IFCA FC*.
[12] Sambuddho Chakravarty et al. 2011. Detecting Traffic Snooping in Tor Using Decoys. In *Proc. RAID*.
[13] David Chaum. 1981. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. ACM* 24, 2.
[14] George Danezis et al. 2003. Mixminion: Design of a Type III Anonymous Remailer Protocol. In *Proc. IEEE S&P*.
[15] Roger Dingledine and Nick Mathewson. 2006. *Tor Path Specification.* github.com/torproject/torspec, accessed: 03/10/2020.
[16] Roger Dingledine et al. 2004. Tor: The Second-Generation Onion Router.
[17] Earn.com. 2013. *Bitcoin Fees for Transactions.* bitcoinfees.earn.com, accessed: 03/10/2020.
[18] Matthew Green and Ian Miers. 2017. Bolt: Anonymous Payment Channels for Decentralized Currencies. In *Proc. ACM CCS*.
[19] Martin Henze et al. 2016. CPPL: Compact Privacy Policy Language. In *Proc. ACM WPES*.
[20] Martin Henze et al. 2014. *A Trust Point-based Security Architecture for Sensor Data in the Cloud.* Springer.
[21] Mirko Knoll et al. 2007. Decentralized Bootstrapping in Pervasive Applications. In *Proc. IEEE PerComW*.
[22] Eleftherios Kokoris-Kogias et al. 2018. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *Proc. IEEE S&P*.
[23] Susan Landau. 2013. Making Sense from Snowden: What's Significant in the NSA Surveillance Revelations. *IEEE Security & Privacy* 11, 4.
[24] Taeho Lee et al. 2018. Bootstrapping Privacy Services in Today's Internet. *ACM SIGCOMM Computer Communication Review* 48, 5.
[25] Brian Neil Levine et al. 2006. *A Survey of Solutions to the Sybil Attack.* Technical Report. University of Massachusetts Amherst.
[26] Isis Lovecruft et al. 2017. *Tor Guard Specification.* github.com/torproject/torspec, accessed: 03/10/2020.
[27] Loi Luu et al. 2016. A Secure Sharding Protocol For Open Blockchains. In *Proc. ACM CCS*.
[28] Roman Matzutt et al. 2018. A Quantitative Analysis of the Impact of Arbitrary Blockchain Content on Bitcoin. In *Proc. IFCA FC*.
[29] Roman Matzutt et al. 2020. How to Securely Prune Bitcoin's Blockchain. In *Proc. IFIP Networking*.
[30] G. Maxwell. 2013. CoinJoin. https://bitcointalk.org/index.php?topic=279249. bitcointalk.org, accessed: 03/10/2020.
[31] Sarah Meiklejohn and Rebekah Mercer. 2018. Möbius: Trustless Tumbling for Transaction Privacy. *PoPETs*.
[32] Sarah Meiklejohn et al. 2013. A Fistful of Bitcoins: Characterizing Payments Among Men with No Names. In *Proc. ACM IMC*.
[33] Prateek Mittal and Nikita Borisov. 2009. ShadowWalker: Peer-to-Peer Anonymous Communication Using Redundant Structured Topologies. In *Proc. ACM CCS*.
[34] Satoshi Nakamoto. 2008. Bitcoin: A Peer-to-Peer Electronic Cash System.
[35] Micha Ober et al. 2013. Structure and Anonymity of the Bitcoin Transaction Graph. *Future Internet* 5, 2.
[36] Andriy Panchenko et al. 2009. NISAN: Network Information Service for Anonymization Networks. In *Proc. ACM CCS*.
[37] Siani Pearson and Marco Casassa Mont. 2011. Sticky Policies: An Approach for Managing Privacy across Multiple Parties. *Computer* 44, 9, 60–68.
[38] Fergal Reid and Martin Harrigan. 2013. *An Analysis of Anonymity in the Bitcoin System.*
[39] Tim Ruffing et al. 2014. CoinShuffle: Practical Decentralized Coin Mixing for Bitcoin. In *Proc. ESORICS*.
[40] Andrey Shevchenko. 2019. *Monero Penalizes GPU and ASIC Mining with RandomX Upgrade.* cryptobriefing.com, accessed: 03/10/2020.
[41] tevador. 2018. *RandomX.* github.com/tevador/RandomX, accessed: 03/10/2020.
[42] The Ethereum Foundation. 2015. *Ethash.* github.com/ethereum/wiki, accessed: 03/10/2020.
[43] The Tor Project. 2007. *Tor Directory Protocol, Version 3.* github.com/torproject/torspec, accessed: 03/10/2020.
[44] The Tor Project. 2009. *Tor Metrics – Relay Search (flag: Authority).* metrics.torproject.org, accessed: 03/10/2020.
[45] The Tor Project. 2014. *Reporting Bad Relays.* trac.torproject.org, accessed: 03/10/2020.
[46] The Tor Project. 2019. *Tor Protocol Specification.* github.com/torproject/torspec, accessed: 03/10/2020.
[47] John Tromp. 2015. Cuckoo Cycle: A Memory Bound Graph-Theoretic Proof-of-Work. In *Proc. IFCA FC*.
[48] Florian Tschorsch and Björn Scheuermann. 2016. Bitcoin and Beyond: A Technical Survey on Decentralized Digital Currencies. *IEEE Commun. Surveys Tuts.* 18, 3.
[49] Marie Vasek and Tyler Moore. 2015. There's No Free Lunch, Even Using Bitcoin: Tracking the Popularity and Profits of Virtual Currency Scams. In *Proc. IFCA FC*.
[50] Philipp Winter et al. 2014. Spoiled Onions: Exposing Malicious Tor Exit Relays. *PoPETs*.
[51] Gavin Wood. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger.
[52] Addy Yeow. 2013. *730-day Charts.* bitnodes.earn.com, accessed: 03/10/2020.
[53] Mahdi Zamani et al. 2018. RapidChain: Scaling Blockchain via Full Sharding. In *Proc. ACM CCS*.
[54] Jan Henrik Ziegeldorf et al. 2015. CoinParty: Secure Multi-Party Mixing of Bitcoins. In *Proc. ACM CODASPY*.
[55] Jan Henrik Ziegeldorf et al. 2018. Secure and Anonymous Decentralized Bitcoin Mixing. *Future Gener. Comput. Syst.* 80.