Increasing Trust in Tor Node List Using Blockchain

Lukáš Hellebrandt[†] Ivan Homoliak[†‡] Kamil Malinka[†] Petr Hanáček[†]

[†] *Department of Intelligent Systems, Faculty of Information Technology, Brno University of Technology*
[‡] *Singapore University of Technology and Design*

***Abstract*—Tor is a low-latency free anonymization network based on onion routing. In Tor, directory servers maintain a list of all nodes. It is, however, possible for a powerful adversary (e.g., law enforcement agencies) to seize or compromise enough directory servers and thus forge that list. Therefore, clients that obtained such a forged list of nodes can be effectively deanonymized. As a countermeasure, we propose to utilize a permissioned blockchain with a single voting committee that is privately "elected" by a verifiable random function (VRF). Since the blockchain provides us with integrity guarantees by design, we increase trust in the directory servers by decentralizing management of Tor nodes present in the shared list. We apply skiplist as an optimization reducing a validation overhead of newly joined nodes and clients. The proposed approach has only a small performance impact on the existing Tor infrastructure.**

I. Introduction

Tor is a low-latency and free anonymization service based on onion routing. Clients of Tor connect to the network and get a list of Tor nodes to build a Tor circuit that provides anonymity. In the past, there were presented several deanonymization attacks – e.g., circuit fingerprinting attacks [1] or linkability of several Tor streams sent over one Tor circuit [2], AS-level routing attacks [18]. However, all of these attacks are orthogonal to our work.

In this paper, we focus on an attack that misuses trust in publicly known directory servers, whose majority can be compromised/seized by the adversary. Directory servers [5] provide a list of Tor nodes on request. There are ten directory servers, and it is enough that the majority of directory servers provide the client with the same list to accept a piece of information contained in the list (i.e., Tor nodes). We assume that it is possible for a strong adversary to compromise the majority (i.e., six) directory servers that feed the client with a forged list of nodes that are controlled by the adversary.

The consequences of this attack are similar as in the case that the client is compromised by an eclipse attack [4], [3] [6] – obtaining a list of Tor nodes that were previously valid but currently are compromised by the attacker, while real directory servers already invalidated them. Under such circumstances, a client builds a Tor circuit that consists only of adversary-controlled nodes, making the anonymization ineffective. This happens because of the adversary who controls all Tor nodes known by a client has the routing information necessary to deanonymize the client. The adversary knows: (1) a source of the message, (2) each node in the Tor circuit used to pass the message, (3) a destination of the message, and optionally even (4) a payload of the message (if no encryption is used).

***Proposed Approach**:* As a countermeasure to this potential vulnerability, we propose to incorporate a permissioned blockchain with a single voting committee into Tor infrastructure. Because each block of the blockchain contains a hash of the previous block, it allows us to check continuity by linking the previous versions of the Tor nodes with the current version. In addition, we propose version tracking of the Tor nodes that serve as peers of the blockchain, while computing their reputation based on identification and geographic location. This setting allows us to distribute trust among all the peers of these blockchains, requiring a consensus of the majority of the committee to add a new peer.

II. Background

A. TOR – The Onion Router

Tor is an anonymization network based on the onion routing principle [10], [11]. In 2019, there are 10 (only 9 until 2018) Tor directory servers worldwide and their public keys and IP addresses are hard-coded into the source code of Tor clients. Directory servers maintain a list of currently active and trusted Tor nodes. A Tor node is a router capable of being a part of the Tor circuit – a set of Tor nodes through which the messages are passed in a defined order, while each node knows only an address of the previous node and the next node in the path.

When a Tor client initiates a new connection to the Tor network, it first builds a Tor circuit. Upon building a Tor circuit, the client gets a list of available nodes from the directory servers. When the information from these servers differs, the correct information is based on the information consistency among the majority of servers [7]. Then, the client selects three (or more) Tor nodes from such a consistent list.

The current architecture of Tor consists of the following components: (1) *Client*, (2) *Tor Nodes*, and (3) *Directory Servers*. The process of creating a new circuit is depicted in Figure 1 and consists of the following steps:[1]

1) The client does an initial connection to the Tor network.
2) The client asks directory servers for a list of Tor nodes.
3) The client receives a list of Tor nodes.
4) The client picks three (or more) arbitrary Tor nodes.

[1]Note that we abstract ourselves from modeling the communication behind Tor nodes, such as hidden services.

978-1-7281-1328-9/19/$31.00 ©2019 IEEE. This work was funded by project VUT IGA FIT-S-17-4014. Next, this work was supported by the NRF, Prime Minister's Office, Singapore, under its National Cybersecurity R&D Programme (Award No. NRF2016NCR-NCR002-028) and administered by the National Cybersecurity R&D Directorate.

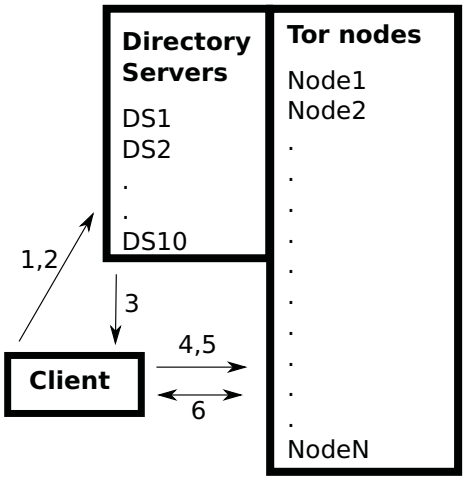

Fig. 1: Current architecture of Tor (simplified).

5) The client builds a Tor circuit from these picked nodes.
6) The client uses the Tor circuit for communication.

B. Blockchain

A blockchain is a linked list of cryptographically connected blocks that can store various data in an append-only fashion – preventing modification of previously stored data and hence ensuring the integrity and immutability of its history. All operations with the blockchain are decentralized among the peers participating in the consensus protocol. In general, there exist permissioned and permissionless blockchains [12]. The former incorporates centralization features into the protocol, partially including the trust, hence only peers approved by a single entity (or federation) may join the consensus (e.g., Hyperledger [16]). In contrast, the permissionless blockchains are decentralized and trust-less, hence any peer may join the consensus protocol immediately (e.g., blockchains with Nakamoto consensus [17]).

Blockchains are also divided [12] according to the election mechanism of the leader/committee that produces a new block into: (1) *single-node consensus* (e.g., Bitcoin [17]), (2) *single-committee consensus* (e.g., Casper the Friendly Finality Gadget [13]), and (3) *multiple-committee consensus* (e.g., sharding [15]). Further, blockchains are divided according to in-band investment to [12]: (1) Proof-of-Work, (2) Proof-of-Stake, (3) Proof-of-Capacity, and (4) Proof-of-Elapsed-Time.

III. Attacks on Tor Directory Servers

Because there are ten directory servers in total, it is required for at least six of them to contain the same piece of information to consider it as trusted. Directory servers are distributed worldwide and run by different parties – only one of them is the *Tor Foundation*. The adversary may be able to compromise/seize six servers. This attack is potentially possible to execute by a powerful adversary (e.g., law enforcement agencies)[2] or by compromising required number of servers.

[2]Note that at the time of writing, four directory servers lay within the same jurisdiction (https://metrics.torproject.org/rs.html#search/flag:authority). Also note that when there were only nine directory servers in 2018, four of them laid within the same jurisdiction.

In the case of such an attack is successful, the list of Tor nodes can be forged to contain the only adversary controlled nodes. This will considerably affect security properties of the whole system. A Tor circuit only consisting of adversary-controlled nodes does not guarantee an anonymity to the client. Therefore, a client that trusts the information from the adversary-controlled list is effectively deanonymized.

IV. Proposed Approach

The proposed architecture is depicted in Figure 2 and consists of the following components: (1) *Client*, (2) *Tor Nodes* (TNs), (3) *Directory servers* (DSes), (4) *Blockchain Peers* (BPs), where

$$BPs = DSes \cup N;\ N \subseteq TNs. \tag{1}$$

In contrast to the current version of Tor, we have added a role of peers that run the blockchain, which consist of directory servers and a subset of the Tor nodes that decided to participate in a consensus protocol of the blockchain.

The peers append two types of transactions that manage a list of the peers and a list of the Tor nodes, respectively. In the following, we detail both transactions types, the consensus mechanism, and finally the steps required to build a Tor circuit with our approach.

A. *Consensus-Level Transactions*

These transactions realize updates to the list of all peers running the blockchain. Therefore, consistency of the current list can be verified with the previously known state, which ensures continuity and makes the attack that forges the list of peers significantly harder. We use directory servers when connecting to the Tor network for the very first time to get a full version of the blockchain and the list of the peers – the consistency of the blockchain is non-interactively validated against (1) the genesis block that is hard-coded in the client and (2) rules of the consensus protocol (see section IV-C). Note that peers are implicitly removed from this list if they are removed from the list of Tor nodes.

B. *Application-Level Transactions*

Data of these transactions update the list of Tor nodes that can be used to build a Tor circuit. The management of this list is done in the same way as in the previous case. Tor nodes can be removed from the list if they are inactive for a certain period of time or considered as compromised (i.e., decisions are subject to the Byzantine fault tolerant (BFT) voting).

C. *Details of Proposed Blockchain*

Since Tor is a free anonymization network running by volunteers, it is not feasible to apply any Proof-of-Work blockchain here – no resources must be wasted. On the other hand, adversarial centralization of consensus power must be made difficult, which is a challenging problem in the setting without incentive mechanisms, such as rewards for voting. Therefore, we propose to apply a public permissioned blockchain with an equal voting power/weight of each peer,

while each peer needs to provide functionality of a Tor node and be approved by reputation-based BFT committee voting to join a protocol. Each peer has its identification (e.g., public IP address) and associated public/private keypair.

When a Tor node wants to join the consensus protocol, it gossips its public key and identification, signed by its private key. Upon receiving a new peer transaction, all peers of the committee validate: (1) the existence of the Tor node using its identification, (2) its correct functionality[3], and (3) the previous invalidation records bound to its identification as well as the geographical location for the purpose of computing a reputation. Note that diversity in geographical locations increases a reputation. Afterward, each peer individually decides whether the new peer is appended to the list of the peers, while at least $\frac{2}{3}+1$ consensus is required to append a corresponding consensus-level transaction to the blockchain. Another message that manages the list of peers is a peer invalidation (e.g., due to unavailability or compromising). When a peer is invalidated, it is not allowed to run the protocol and to join the protocol again, it has to obtain approval from $\frac{2}{3}M+1$ members of some committee, subject to an assessment of the reputation.[4]

Consensus within a Committee: The consensus among peers starts by selecting a single committee that consists of M members. The committee is selected based on a lottery approach – each participant runs a verifiable random function (VRF) [14] with input consisting of a private key, a random seed from the previous block, and the height of the block (i.e., input depends only on the blockchain state). Then each peer privately observes whether the output is under a certain threshold T,[5] and if so, the peer is *a candidate for a committee member* and it must broadcast (by gossiping) the value of VRF and all its votes (i.e., transactions) related to the round. Under the model of eventual network synchrony, each peer selects M nodes that provided the lowest value of VRF to form a committee in a non-interactive fashion – i.e., the votes were already cast, so each node can determine which transactions were approved by at least $\frac{2}{3}M+1$ of known committee members. Afterward, transactions are unambiguously ordered, the lowest observed VRF output is used as a random seed of the block, and a new block is created and gossiped to the network. Note that if a peer receives a block with a higher or the same value of block's random seed than a peer is aware of, the propagation of that block is stopped.

If there are no online committee members – no valid transactions are received within a specific time frame (e.g., one hour), an empty block is appended by each peer, causing a reset of the random seed by computing a hash of the previous seed, which in turn increases a chance that a new committee will contain online nodes. Then a peer gossips the block to all peers, who upon receiving the block validate: (1) block header – i.e., binding to the previous block header, (2) correctness of all signatures present in the block, and (3) a sufficient number of signatures in each transaction of the block.

[3]Note that validation of the functionality is done only by a small number of peers, and it does not cause a DoS attack on the target node.

[4]Note that directory servers cannot be invalidated and also note that nodes may change their identification (e.g., IP addresses); however, we assume that this requires resources spent.

[5]T is the network adjusted parameter.

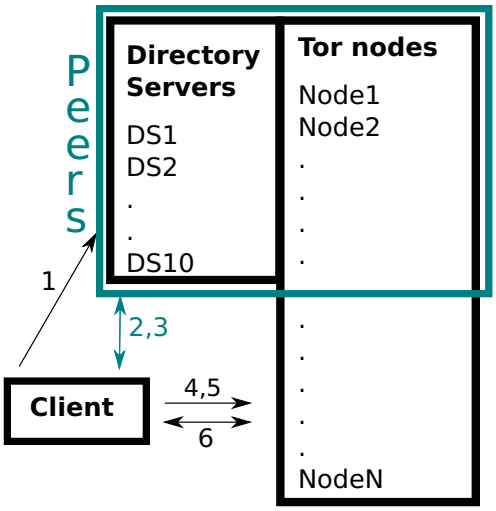

Fig. 2: Proposed architecture of peers that are participating the consensus of the blockchain (depicted in a green frame).

Check-Pointing: In order to avoid reverting the full blockchain history by adversaries that temporarily control the majority of the consensus power (i.e., 51% attack), we propose check-pointing of blocks once per day (e.g., every 24 blocks), accomplished by a present committee.

Fork Choice Rule – the Strongest Chain: When the client decides whether to trust in a particular chain and adapt it, the strongest chain rule is applied, which is a standard way in other existing blockchains. In our approach, the strongest chain rule selects the block with the smallest value of random seed. This rule allows for changing the history up to the most recent checkpoint. Although this does not protect us from 51% attack between two consecutive checkpoints, it provides us with a relatively high level of trust regardless of whether the attack was done previously or is currently being held:

- If the attack occurred in the past and the blockchain has been fixed already, then the client is safe, as it obtained the most recent version of the blockchain.
- If the attack is currently being held by an attacker strong enough to seize most of the consensus power (or only the of directory servers if the client connects for the first time), then the client will temporarily use a forged list of nodes – resulting into deanonymization in that particular session. However, the blockchain will be eventually fixed (by flipping the majority of peers to be honest) and the client's next connections will be anonymous. The honest node can "repair" the current version of the blockchain within two checkpoints. Nevertheless, repairing the blockchain after the next checkpoint would cause a hard fork, and it would require newly joined users to always use an updated version of Tor clients.

D. *Building a Tor Circuit*

With our proposed approach, the process of building a Tor circuit works as follows (see Figure 2):

1) The client makes an initial connection to the network
2) The client downloads and validates the blockchain (or its missing part) from several randomly selected known valid peers – if peers provide different blockchains, then the client randomly selects other peers, until at least $\frac{2}{3}M+1$ of peers provide the consistent version.
3) The client can optionally repeat the previous step using peers from the consistent part of the newly added blocks.
4) The client randomly picks three (or more) nodes from the actual list of valid Tor nodes.
5) The client builds a Tor circuit from the picked nodes.
6) The client communicates using the Tor circuit.

E. *Optimizations*

Although transactions and blocks of our approach are sparse, each new client and Tor node joining the protocol must download and validate the full blockchain, which may take a while mainly due to several signature verifications per each transaction in a block. Therefore, we propose to use forward links of skiplist [19] with witness co-signing protocol [20] (Co-Si) run within a committee. Co-Si with skiplist retrospectively builds forward links from the past to the future blocks, upon creating the current blocks (as made in Chainiac [21]). This enables new client and Tor nodes to skip signature validation overhead after downloading the blockchain while relying on committees that co-signed the forward links.

V. Discussion

Blockchain, in general, allows us for checking integrity and continuity, as it is an append-only data structure, in which each block is cryptographically linked to the previous one. The application of permissioned blockchain with reputation assessment allows us to trust more in the list of Tor nodes and the list of peers they contain. This is because to significantly influence the protocol, the adversary would have to control more than a half of the peers (51% attack), which is protected by reputation-based voting that also favors diverse geographic locations. On the other hand, using a single committee for reaching the consensus requires $\frac{2}{3}M+1$ peers to vote within specified time-frame, which may be a potential issue when more than $\frac{1}{3}M$ peers are not voting (either intentionally or they are temporarily disconnected). To cope with this issue, an empty block is appended by each peer, which resets the committee. Since the transaction and blocks of the proposed approach are sparse, the empty blocks do not deteriorate the performance of the protocol and consume only a minimal space.

VI. Conclusion

In this paper, we proposed to use the permissioned blockchains distributed among some Tor nodes, serving as blockchain peers. The blockchain holds and versions a list of its valid peers as well as the list of all Tor nodes. This allows a Tor client to trust more in the validity of these lists and the information present in them. Thanks to the properties of blockchains, and decentralization, in particular, we believe that our approach provides a higher level of trust in Tor infrastructure in contrast to the current state (see subsection II-A). In future work, we plan to evaluate our approach by simulation experiments and make a proof-of-concept implementation.

References

[1] Kwon, A., AlSabah, M., Lazar, D., Dacier, M. and Devadas, S., 2015, August. Circuit fingerprinting attacks: Passive deanonymization of tor hidden services. In 24th USENIX Security Symposium.

[2] Blond, S.L., Manils, P., Abdelberi, C., Kaafar, M.A.D., Castelluccia, C., Legout, A. and Dabbous, W., 2011. One bad apple spoils the bunch: exploiting P2P applications to trace and profile Tor users. arXiv preprint arXiv:1103.1518.

[3] A. Singh, "Eclipse attacks on overlay networks: Threats and defenses" in IEEE INFOCOM, 2006.

[4] E. Heilman and A. Kendler, "Eclipse Attacks on Bitcoin's Peer-to-Peer Network," in USENIX Security Symposium 2015 (pp. 129-144).

[5] I. Ahmad, M. Saddique, U. Pirzada, M. Zohaib, A. Ali and M. Khan, "A New Look at the Tor Anonymous Communication System" in Journal of Digital Information Management, 16(5), p.223., 2018

[6] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang and Z. H. Tian, "Towards a Comprehensive Insight into the Eclipse Attacks of Tor Hidden Services" in IEEE Internet of Things Journal, 2018.

[7] torproject.org, "Tor Metrics - Glossary", 2018. [Online]. Available: https://metrics.torproject.org/glossary.html#consensus

[8] A. Gervais, G. O. Karame, K. Wüst, V. Glykantzis, H. Ritzdorf and S. Capkun, "On the security and performance of proof of work blockchains" in Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (pp. 3-16), October 2016

[9] I. C. Lin, and T. C. Liao, "A Survey of Blockchain Security Issues and Challenges" in IJ Network Security, 19(5), pp.653-659, 2017

[10] P. Syverson, R. Dingledine and N. Mathewson, "Tor: The second generation onion router" in Usenix Security, 2004

[11] D. Goldschlag, M. Reed, P. Syverson, "Onion routing" in Communications of the ACM, 1;42(2):39-41, Feb 1999

[12] S. Bano, A. Sonnino, M. Al-Bassam, S. Azouvi, P. McCorry, S. Meiklejohn, and G. Danezis, "Consensus in the age of blockchains" in arXiv preprint arXiv:1711.03936, 2017

[13] V. Buterin, and V. Griffith, "Casper the friendly finality gadget" in arXiv preprint arXiv:1710.09437, 2017

[14] Gilad Y, Hemo R, Micali S, Vlachos G, Zeldovich N. Algorand: Scaling byzantine agreements for cryptocurrencies. InProceedings of the 26th Symposium on Operating Systems Principles 2017 Oct 14 (pp. 51-68). ACM.

[15] Fynn, E. and Pedone, F., 2018. Challenges and pitfalls of partitioning blockchains. arXiv preprint arXiv:1804.07356.

[16] Hyperledger, "Hyperledger architecture, volume 1: Consensus," 2017. [Online]. Available: https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger

[17] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[18] Sun Y, Edmundson A, Vanbever L, Li O, Rexford J, Chiang M, Mittal P. RAPTOR: Routing Attacks on Privacy in Tor. In24th USENIX Security Symposium (USENIX Security 15) 2015 (pp. 271-286).

[19] Munro JI, Papadakis T, Sedgewick R. Deterministic skip lists. In Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms 1992 Sep 1 (pp. 367-375). Society for Industrial and Applied Mathematics.

[20] Syta E, Tamas I, Visher D, Wolinsky DI, Jovanovic P, Gasser L, Gailly N, Khoffi I, Ford B. Keeping authorities" honest or bust" with decentralized witness cosigning. In2016 IEEE Symposium on Security and Privacy (SP) 2016 May 22 (pp. 526-545).

[21] Nikitin K, Kokoris-Kogias E, Jovanovic P, Gailly N, Gasser L, Khoffi I, Cappos J, Ford B. CHAINIAC: Proactive Software-Update Transparency via Collectively Signed Skipchains and Verified Builds. In26th USENIX Security Symposium (USENIX Security 17) 2017 (pp. 1271-1287).