TorCoin

Bryan Ford, Miles Richardson, and Mainak Ghosh

Yale University, New Haven, CT
{bryan.ford, miles.richardson, mainak.ghosh}@yale.edu

Abstract. The scalability of the Tor Anonymity Network suffers from the lack of an incentive to volunteer bandwidth resources, which are required for the system to operate effectively. This paper identifies the challenges and issues involved in producing such incentives, and proposes an architectural design with acceptable trade-offs that can be realized with mostly existing technologies.

1 Introduction

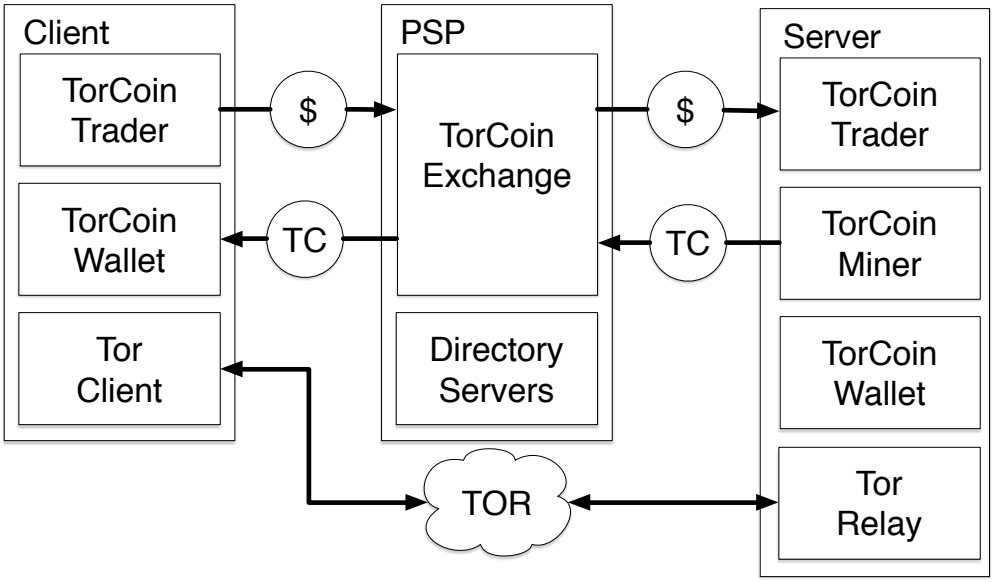
The Tor network suffers from slow speeds, because of a shortage of relay nodes, unsustainably provided by altruistic volunteers. This is a well studied problem. Despite many attempts, researchers have failed to deploy a widely-adopted mechanism for compensating relay operators, while retaining anonymity of clients. [1–7].

First, we propose a radical rethinking of the economic structure of the Tor anomymity network. Specifically, we introduce the idea of *privacy service providers* (PSPs), analagous to internet service providers (ISPs) in current nomenclature. A PSP operates an independent Tor network. It charges clients for access, and pays relay operators per-megabit of bandwidth transferred.

Second, we present an anonyimity-preserving software architecture to enable this economic model, without modifying the core codebase of Tor. We introduce '*TorCoin*', an alternative cryptocurrency (altcoin) that relay operators can mine by transferring bandwidth. It uses the BitCoin protocol, but relies on proof-of-bandwidth instead of proof-of-computation.

TorCoin enables the economic model to work. Relay operators install a *TorCoin Miner*, which verifiably mints TorCoins per-mb of transfer. They also install a *TorCoin Trader*, which sells their TorCoins on a *TorCoin Exchange* for cash. Each PSP operates an exchange; clients pay cash to access the network, and the PSP pays cash to relays for TorCoins.

2 Economic Architecture



3 Software Architecture

We now discuss the main components necessary to realize a practical Tor incentive system while identifying some open research and development problems.

3.1 Overview

We propose a system that measures bandwidth contributed to the Tor network to produce incentivise the addition of nodes to the Tor network.

The system measures the bandwidth contributed by each relay in the Tor network and rewards them with a 'TorCoin'. A Torcoin is an AltCoin that uses a bandwidth-intensive protocol as its proof-of-work. Thus, to produce a TorCoin, a relay must have transmitted a certain amount of Tor traffic.

To reduce the system's vulnerability to attackers and possible reduction of anonymity, we also utilize a system of 'Ephemeral Paths' to randomly assign relays to clients.

These TorCoins can then be traded at an exchange for other AltCoins or other goods. This forms the basis of our incentivization scheme. This is different from systems that propose differentiated service [8, 9], since we do not propose to make the clients pay for access to the network. The coins are a byproduct of the usage of the system.

3.2 Ephemeral Paths

We propose the novel idea of ephemeral Tor networks, where a *group* of n clients at any given time form a consensus on an assignment of n routes to n clients,

such that each route is publicly verifiable but privately addressable. We adhere to the following constraints:

- No client in the group can generate its own route.
- Every resulting route has a unique public key.
- No client in the group can know the route assigned to another client in the group.
- Any client can verify that a given public key represents a route assigned to a client in the current group.

The Tor directory servers will create the groups using the temporal locality of the clients connecting to them, but also ensure that there is geographical and other diversity in a group. This is to ensure that adversaries cannot deterministically place themselves in a single group by connecting at the same time.

The Tor directory server will then initiate the setup protocol amongst the clients in each group. Each group will then perform a series of Neff Shuffles to produce a random combination of routes for each client in the group. This will be combined with El Gamal encryption to preserve anonymity within the group.

Decentralized Neff Shuffle for Consensus Tor Route Assignment to n Clients IFFY BIT - PLEASE PAY ATTENTION We perform a series of Neff shuffles so that each client in the group is assigned one *route* consisting of an entry, middle and exit relay. The route also has a publicly verifiable shared secret. That is, a route can collectively sign a TorCoin (explained later) such that any client in the group can verify that the signature came from a route assigned to another client in the group.

Implementation of Neff Shuffle THIS SECTION NEEDS TO BE MADE CLEARER / DIAGRAM

The clients are numbered 1 through N

Entry relays 1 to Me;

Middle relays: 1 through Mm;

Exit relays: 1 through Mx.

1. Each list of relays and clients goes through an independent Neff shuffle. The shuffled lists are concatenated, producing a shuffled 4xN matrix. Each client will have one row assigned to it, which only that client can verifiably associate with. This row represents the route for that client.

2. Each client can use their own private key to find out their own row and assigned relays after the shuffle, but no one else can.

3. The directory server generates a public and private key for all members of the row. This functions as the shared secret for the route.

It may be possible to use cryptographic accumulators along with the Neff shuffle. In that case, it will not be necessary to keep all the public keys generated in part 3. Instead, the keys may be used to generate a cryptographic accumulator and this accumulator can be used to provide a zero-knowledge verification of group-membership.

We can then generate a publicly verifiable shared secret amongst the relays in any given route (a row in the 4XN matrix). That is, a route can collectively

sign a TorCoin such that any client in the consensus group can verify that the signature came from a route assigned to another client in the consensus group.

IFFY BITS

The servers come together to create groups. They use Neff shuffles to create shuffled lists of Entry, middle and exit relays. They also shuffle a list of clients in the group. They then create a route-shared-key for each route based on some property of each participant (client and three relays). Each route-shared-key is also input into an accumulator so that anyone can verify if a route belonged to a group.

The servers then inform the clients of [entry, middle, exit, route shared key, accumulator] and the relays are informed of their Access Control List (list of all the clients and relays they should accept connections from.)

Problems: PSP has total information about who is connecting to whom. Prof. Ford probably has some idea about how to obfuscate this.

Proof of Bandwidth - Onion Hashing Once the routes are setup, we can prove bandwidth transfer through the following protocol: Every n Tor packets, the client sends an extra packet (the Torcoin packet) containing an hash attempt likely to generate a TorCoin. Relay A gets it, generates a temporary private key Ka (generated using the route shared key) and hashes the received packet and this key. It then forwards it to B, which does the same thing, with its own private temporary key Kb. Similarly on to C. C can now add its own Kc, and if it generates a hash with a given number of zeros, it can claim to have found a TorCoin.

```
Client sends to A: T0 (its hash attempt)
A sends to B     : Hash(T0 + K1) = Ta # K1 is A's temporary private key.
B sends to C     : Hash(Ta + K2) = Tb # K2 is Bs temporary private key.
C computes       : Hash(Tb + K3) = Tc # K3 is Cs temporary private key.
C sends to B     : (Tc, K3) to verify.
B sends to A     : (Tc, K3, Tb, K2) to verify.
A sends to client: (Tc, K3, Tb, K2, Ta, K1) to verify.
Once the client has verified the hash, we can confirm that the data has made a
round trip through the route. This completes the proof of bandwidth.
```

TorCoin We can then implement an AltCoin based on the proof-of-work concept in the following manner:

```
If (Tc = '000...')
  If the client succesfully verifies the hash
    It adds the coin to the blockchain with the following information:
    1. Clients public key
    2. Route Shared key (Lets any other group member verify that the route is
       genuine. See accumulator.)
    3. Accumulator for the group.
    4. TorCoin Hash
    It then gives 1/3rd of the coin to each of the relays in the route. (If the
    client is rogue, can we identify and kick the client off?)
```

3.3 Robustness to attack

The entire reason for constructing the elaborate ephemeral routes algorithm is to make the Torcoin system robust to attackers.

Due to the random group selection system, it is hard for attackers to deterministically place themselves in a group. In addition, because the attacker needs to control all four components fo a route to mint a TorCoin fraudulently, even if the adversaries control up to half the network, there is a probability of only 1/16 that an adversary client gets a path of three colluding relays. In practice, gaining control of half of the entire Tor client and relay network is practically impossible.

A separate rate-limiting mechanism can be deployed to detect dishonest relays and assign them a lower weightage in the path selection procedure. An independent verification authority, such as one based on Eigenspeed, could be used to detect these discrepancies.

4 Preliminary Results

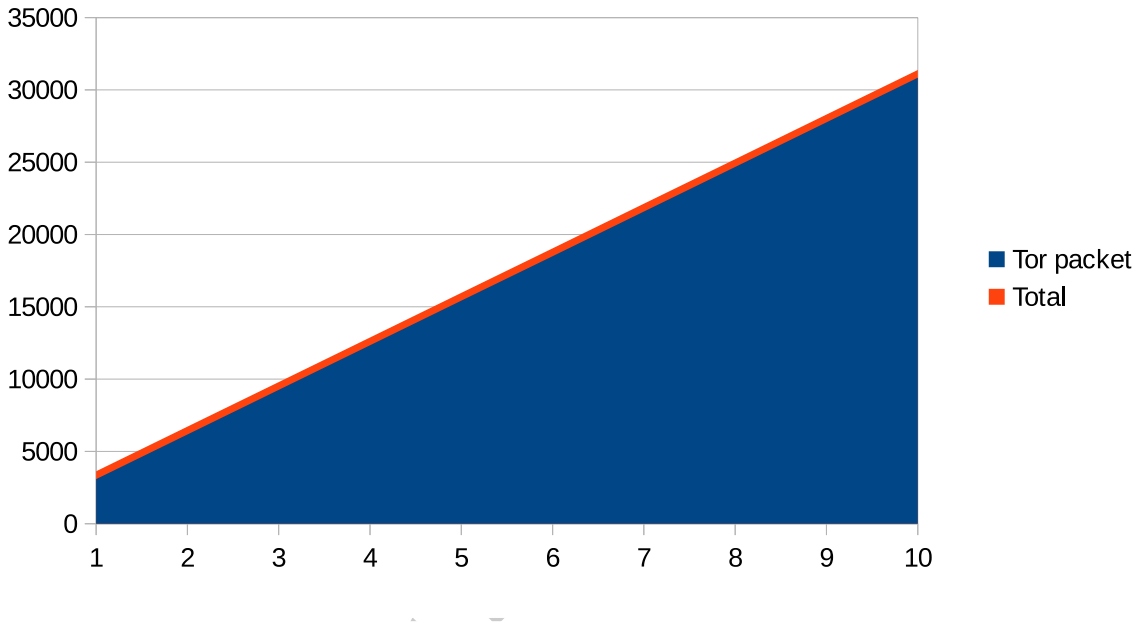
The TorCoin protocol does add a small amount of overhead to the Tor traffic. In our experimental setup, we set up a series of servers using the Python Twisted framework to simulate the passing of TorCoin generation and verification messages through a set of relays.

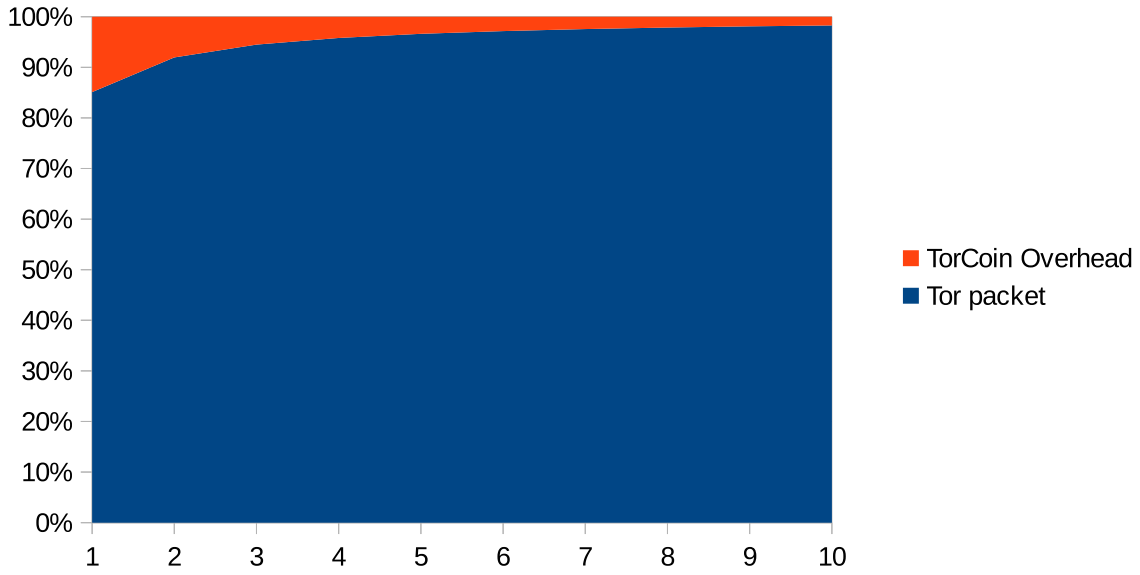
The total overhead from one round of successful TorCoin mining (i.e., one entire round trip from client through all the relays and back again) results in a total TorCoin packet overhead of 540 bytes. This can be broken down into:

- The initial three attempts by each relay: 42 bytes
- Verification message from Relay 3 to Relay 2: 74 bytes
- Verification message from Relay 3 to Relay 2: 138 bytes
- Verification message from Relay 3 to Relay 2: 202 bytes

Each round of TorCoin generation and verification happens only after n Tor packets have been sent. Each standard Tor cell is 514 bytes long, so each round trip on the network requires transmission of 514 * 6 = 3084 bytes. Thus, if n ¿ 10, the TorCoin protocol overhead is less than 2%. The value of n can be calibrated in further experimentation and as needed in order to achieve the sweet-spot of transmission efficiency and incentive maximization for relay providers.

Indeed, one can envision a tunable system where the value of n decreases during times of high usage to incentivise relay providers to temporarily provide more servers to the network.





While the Neff shuffle is complicated and requires a large number of communications between all of the servers, in practice, since the number of directory servers that will be involved in each shuffle will be relatively small (less than 10)

and are relatively fast servers with high-bandwidth connections with each other, this will not be a major bottleneck. In addition, since this is a one-time cost of connecting to the network, the users will be willing to wait for the slight time that it takes to setup the protocol, especially if the speeds increase sufficiently.

5 Related Work

PAR [1], XPay [2], Gold Star [3], BRAIDS [4], Tortoise [5], LIRA [6], onions for sale [7].

On the economics of anonymity [10], one-to-n scrip systems [11].

6 Future Work and Conclusions

References

1. Androulaki, E., Raykova, M., Srivatsan, S., Stavrou, A., Bellovin, S.M.: PAR: Payment for anonymous routing. In Borisov, N., Goldberg, I., eds.: Privacy Enhancing Technologies: 8th International Symposium, PETS 2008, Leuven, Belgium, Springer-Verlag, LNCS 5134 (July 2008) 219–236
2. Chen, Y., Sion, R., Carbunar, B.: XPay: Practical anonymous payments for Tor routing and other networked services. In: Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2009), ACM (November 2009)
3. Ngan, T.W.J., Dingledine, R., Wallach, D.S.: Building incentives into Tor. In Sion, R., ed.: Proceedings of Financial Cryptography (FC '10). (January 2010)
4. Jansen, R., Hopper, N., Kim, Y.: Recruiting new Tor relays with BRAIDS. In Keromytis, A.D., Shmatikov, V., eds.: Proceedings of the 2010 ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010, ACM (2010)
5. Moore, W.B., Wacek, C., Sherr, M.: Exploring the potential benefits of expanded rate limiting in tor: Slow and steady wins the race with tortoise. In: Proceedings of 2011 Annual Computer Security Applications Conference (ACSAC'11), Orlando, FL, USA. (December 2011)
6. Jansen, R., Johnson, A., Syverson, P.: LIRA: Lightweight Incentivized Routing for Anonymity. In: Proceedings of the Network and Distributed System Security Symposium - NDSS'13, Internet Society (February 2013)
7. Johnson, A., Jansen, R., Syverson, P.: Onions for sale: Putting privacy on the market. In: Financial Cryptography and Data Security. Springer (2013) 399–400
8. Dovrolis, C., Ramanathan, P.: A case for relative differentiated services and the proportional differentiation model. Network, IEEE **13**(5) (1999) 26–34
9. Dovrolis, C., Stiliadis, D., Ramanathan, P.: Proportional differentiated services: Delay differentiation and packet scheduling. IEEE/ACM Transactions on Networking (TON) **10**(1) (2002) 12–26
10. Acquisti, A., Dingledine, R., Syverson, P.: On the economics of anonymity. In: Financial Cryptography. Springer-Verlag, LNCS 2742. (2003) 84–102
11. Humbert, M., Manshaei, M., Hubaux, J.P.: One-to-n scrip systems for cooperative privacy-enhancing technologies. In: Proceedings of the 49th Annual Allerton Conference on Communication, Control, and Computing. (2011)