A TorPath to TorCoin:
Proof-of-Bandwidth Altcoins for Compensating Relays

Bryan Ford, Mainak Ghosh, and Miles Richardson

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

Abstract. The Tor network relies on volunteer relay operators for relay bandwidth, which may limit its growth and scaling potential. We propose an incentive scheme for Tor relying on two novel concepts. We introduce *TorCoin*, an "altcoin" that uses the BitCoin protocol to reward relays for contributing bandwidth. Relays "mine" TorCoins, then sell them for cash on any existing altcoin exchange. To verify that a given TorCoin represents actual bandwidth transferred, we introduce *TorPath*, a decentralized protocol for assigning Tor circuits to clients such that each circuit is privately-addressable but publicly verifiable. The participants on each such circuit may then collectively mine a limited number of TorCoins, in proportion to the end-to-end transmission goodput they measure on that circuit.

1 Introduction

The Tor network suffers from slow speeds, due to a shortage of relay nodes from volunteers. This is a well studied problem, but despite many attempts, there is not yet a widely-adopted mechanism for compensating relay operators while retaining anonymity of clients [1–7]. We present a possible solution, embodying two complementary novel ideas:

1. **TorCoin** is an alternative cryptocurrency, or altcoin, based on the BitCoin protocol [8]. Unlike BitCoin, its proof-of-work scheme is bandwidth-intensive, rather than computationally-intensive. To "mine" a TorCoin, a relay transfers bandwidth over the Tor network. Since relays can sell TorCoin on any existing altcoin exchange, TorCoin effectively compensates them for contributing bandwidth to the network, and does not require clients to pay for access to it.

2. **TorPath** is a method for decentralized groups of "Assignment Servers," which extend traditional "Directory Servers," to assign each client a Tor circuit that is publicly verifiable, but privately addressable. This mechanism allows TorPath to "sign" newly-minted TorCoins, so that anyone can verify a TorCoin by checking the blockchain.

2 Motivation and Related Works

Solving the problem of compensating Tor relays is attractive because it would immediately improve the scalability of the Tor network. Prior research has not arrived at a satisfactory solution that operates within the basic requirements of an incentive scheme. We now outline those requirements.

2.1 Requirements of an Incentive System

At a basic level, an incentive system must retain anonymity but have the ability to verifiably measure bandwidth and reliably distribute payment to the nodes that provide it. The system must be resilient to adversaries attempting to identify clients, or fake bandwidth transfer.

Anonymity-Preserving Architecture TorCoin should preserve anonymity. Currently, no Tor client can recognize another, and no relay can identify the source or destination of any packet it transfers. Proposed incentive schemes like Tortoise [5] and Gold Star [3] have compromised clients' anonymity by allowing their traffic to be identified [6]. In a proportionally differentiated service [9, 10] like LIRA [6] a speed-monitoring adversary can potentially partition the anonymity set into clients that are paying for higher speeds, thus reducing anonymity. An acceptable implementation of TorCoin will at least retain the anonymity of the current Tor protocol, but an excellent one will improve upon it.

Verifiable Bandwidth Accounting TorCoin needs to measure bandwidth in such a way that anyone can verify its measurements. Optimally, it will not require self-reporting or centralized servers, unlike EigenSpeed [11] or Opportunistic Bandwidth Monitoring [12]. The system should be robust to attackers or groups of attackers colluding to misreport bandwidth measurements, and the entire network should agree on all measurements. Rather than relying on reported network speeds, TorCoin uses an onion-hashing scheme to push "TorCoin Packets" through the circuit to measure its end-to-end throughput.

Anonymity Preserving Payment Distribution Once TorCoin measures the bandwidth of a given node, it must distribute payment to that node in a way that preserves anonymity. Specifically, it should not be possible for anyone to associate a bandwidth payment or measurement with a specific relay. Since TorCoin also requires verifiable accounting, the problem becomes how to verify bandwidth without identifying its provider.

Highly-Available, Reliable Transaction Storage TorCoin must store sufficient records of previous payments to avoid rewarding a relay twice for the same bandwidth transfer. Other proposed Tor incentive schemes like LIRA [6] use a trusted central bank to assign coins and track spending. This makes the entire incentive scheme dependent on the central authority. Instead, TorCoin uses the BitCoin protocol's distributed ledger to track transactions and avoid double spending [13]. There is no central authority in-charge of the incentive system.

Scalable and Deployable with Minimal Code Changes To simplify deployability, TorCoin should operate mostly as a wrapper around the existing Tor clients. It should require minimal changes to the core Tor codebase, and should not significantly increase latency of requests. A good incentive scheme should also be scalable to accommodate a large number of users and relays operating concurrently. Unlike schemes like BRAIDS [4], TorCoin is designed to be scalable because of its decentralized structure, small transaction overheads and use of protocols like Bitcoin to implement some features.

2.2 Key Technical Challenges

We might envision a naive bandwidth-measurement scheme using blinded cryptographic tokens to signify bandwidth transfer. Suppose in this scheme, a client is able to give each relay a token for the amount of bandwidth it provides. Relays are then able to convert these client-signed tokens into some form of an incentive. Such a scheme would be vulnerable to colluding groups of clients and relays, however, who can simply sign each other's transfer tokens without actually transferring any bandwidth.

We attempt to counter this through the TorPath scheme, which restricts clients' ability to choose their own path, ensuring that *most* paths include at least one non-colluding participant (the client or at least one of the three relays). Assignment servers bundle large groups of clients and relays into *groups* that collectively choose paths. Even in the relatively rare event that a path constructed this way consists entirely of colluding clients and relays, an upper bound on the number of coins each path can mint rate-limits potential loss to such entirely-colluding paths.

3 Proposed Architecture

3.1 Overview

The TorCoin architecture implements the novel TorCoin and TorPath protocols. We describe both in later sections, but in brief, the TorCoin protocol mines coins, and the TorPath protocol assigns a circuit (entry, middle, and exit servers) to each client.

TorCoin runs as a standalone service, and requires little modification of the core Tor codebase. Tor clients and relays operate as usual, but receive circuit assignments from *assignment servers* instead of directory servers. Separately, a *TorCoin Miner* on each machine mines TorCoins by monitoring the throughput of the local Tor TLS tunnel, and communicating with its circuit neighbors via the TorCoin algorithm.

Figure 1 shows a basic overview of this architecture.

3.2 Adversary Model

We consider the existence of an adversary who wishes to obtain TorCoins without trnsferring bandwidth. We assume the adversary is able to control a number of clients and routers. We assume that malicious clients and routers know about each other's existence and are able to collude. We also assume that the adversary

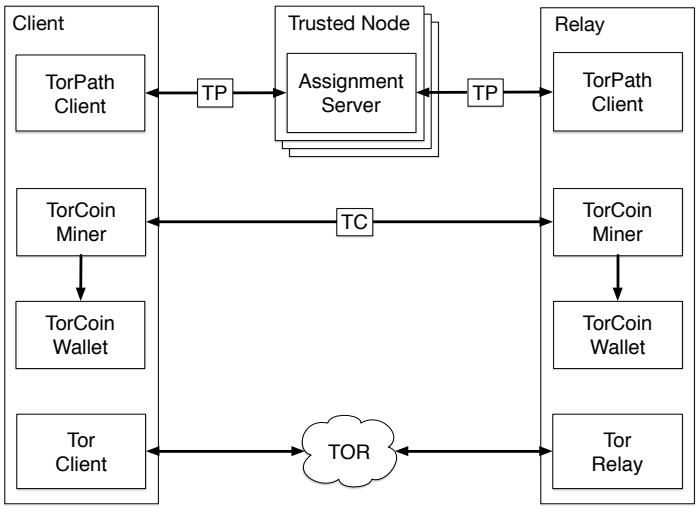

Fig. 1. High level TorCoin system architecture for clients and relays. A *TorPath Client* assigns Tor circuits to clients via the TorPath protocol, described in the next section. A *TorCoin Miner* mines TorCoins and stores them in a *TorCoin Wallet*. Each *Tor Client* and *Tor Relay* operates as usual, but on circuits assigned via the TorPath protocol.

is able to control a minority of assignment servers on the network and that the other servers are honest-but-curious.

3.3 The TorPath Protocol

Overview The TorPath protocol assigns Tor circuits to clients, overriding Tor directory servers with *assignment servers*, which form decentralized *consensus groups*. The protocol guarantees that no participant on a circuit can identify all other participants, and that each circuit includes a publicly verifiable signature. We use TorPath to "sign" each TorCoin, so that anyone can verify a TorCoin by comparing its signature to a global circuit history.

Requirements The TorPath protocol adheres to the following constraints:

- No client can generate its own circuit.
- Every circuit has a unique, publicly-verifiable signature.
- No client can know the circuit of another client.

Protocol Description The protocol consists of three sequential steps:

1. **Group Initialization.** Assignment servers form a *consensus group*. Clients and relays provide public keys to join the group.
2. **Verifiable Shuffle.** The consensus group performs a decentralized, verifiable shuffle of all the public keys, resulting in a circuit assignment for each client.
3. **Path Lookup.** The assignment servers publish the result of the shuffle, such that each client can only identify its entry relay, and each relay can only identify its immediate neighbor(s).

(1/3) Group Initialization A consensus group is formed when a majority of the Assignment servers come together to assign circuits to clients. Thus, if there are 10 Assignment Servers in the network, at least 6 of them must collectively form a group. The size of each consensus group can be modulated to be a number n, i.e., the group waits until there are n clients to proceed to the next stage. Different groups can have different values of n to allow each client to choose its own anonymity level. Groups with larger values of n would provide a larger anonymity set, at the expense of longer circuit setup times. When the group waits to reach its threshold number n of connected clients before it moves on to the next step.

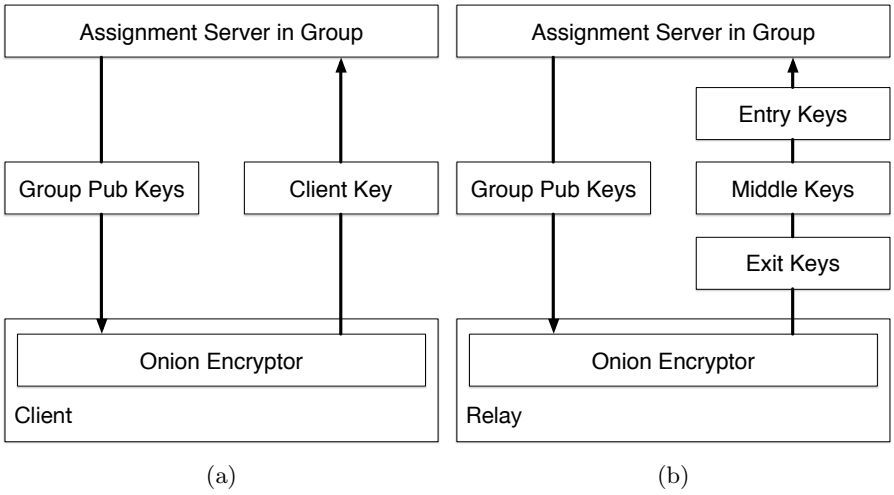

Fig. 2. Both relays and clients use onion-encryption to encrypt their own temporary public keys with the public keys of all the assignment servers in the group. (a) Client generates one keypair. The public key is sent onion-encrypted to the server. (b) Each relay generates multiple keypairs (for different clients and positions) as instructed by the assignment servers.

A consensus group forms in three separate steps:

1. **Each assignment server** shares its public key with its group members, and broadcasts the public keys it receives to all clients and relays connected to it.
2. **Each client** connects to one assignment server in the group. The client then generates a temporary private and public key pair. It uses "onion encryption" to combine its public key with the public keys of all assignment servers in the group, resulting in a hash that each server can only partially decrypt. It sends this hash to its own server.

3. **Each relay** can act as an entry, middle, and/or exit relay, and it chooses which position(s) to assume. In most cases, the number of available relays will be less than that of clients in a group. To ensure parity between clients and relays for each position, each assignment server instructs its relays to generate a sufficient number of keys for each position. The relay server uses onion-encryption to generate n cipher-texts from n temporary public keys. It packages them by position and sends them to its assignment server.

Figure 2 illustrates Steps 2 and 3, after the assignment servers exchange public keys.

We can represent the list of keys as a $4 \times n$ matrix M for n clients, where each row corresponds to a client and its three relays (see Figure 3).

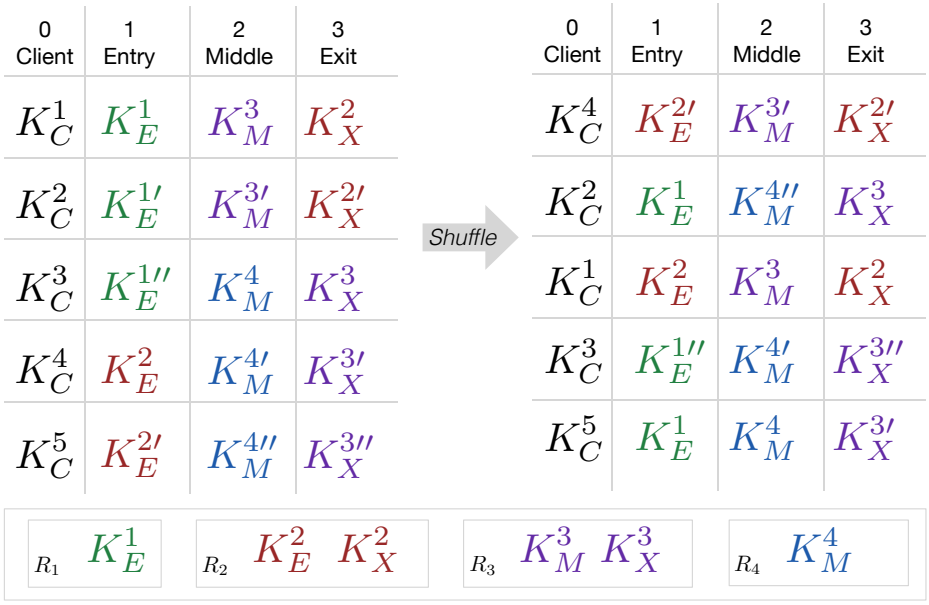

Fig. 3. Example matrix shuffle with 5 clients (C_1, C_2, C_3, C_4 and C_5) and 4 relays (purple, blue, green, red). Each relay R_i generates a number of public keys K_p^i for each circuit position $p \in E, M, X$, as instructed by its assignment server. Here, each client C_j is assigned the circuit represented by the j^{th} row of the shuffled matrix.

(2/3) Verifiable Shuffle The consensus group then "shuffles" each column (using a verifiable shuffling algorithm, like the Neff Shuffle [14]) so that each the row for each path contains a random 4-tuple of relay public keys (the client and the three relays serving that client).

The assignment servers collectively sign and publish the resulting matrix to a public log, accessible by all clients and relays.

(3/3) Path Lookup Path lookup is the final step of the algorithm, where each client obtains the IP address of its entry relay, and each relay obtains the IP address of its neighbor(s) on the circuit. The path lookup algorithm ensures that each client and relay can only receive the IP address of its circuit neighbors.

Each client encrypts its own IP using the public key of its neighbor. It then creates a tuple of the form (public key, encrypted IP) as described in table 1. This tuple is then onion-encrypted and sent to the server. Each relay also follows the same procedure, but for every key in the matrix that belongs to it.

The assignment servers shuffle this new list of tuples. At this point, each client and relay can find its neighbors in the matrix by locating the tuples containing the public keys it needs. Finally, it decrypts those cells using its private key, revealing the IP address of its circuit neighbor(s).

Now clients have a circuit they can use for Tor.

Sender	**Message Tuple**
Client	$(K_C^i, \{IP_C^i\}_{K_E^i})$
Entry relay	$(K_E^i, \{IP_E^i\}_{K_C^i}, \{IP_E^i\}_{K_M^i})$
Middle relay	$(K_M^i, \{IP_M^i\}_{K_E^i}, \{IP_M^i\}_{K_X^i})$
Exit relay	$(K_X^i, \{IP_X^i\}_{K_M^i})$

Table 1. Each participant in a circuit sends its message tuple onion-encrypted to the server. $\{X\}_Y$ denotes X encrypted with Y.

Circuit Signature The Circuit Signature of the i^{th} circuit in a shuffle is an ordered pair consisting of the hash of the matrix M and the row number of the circuit in the matrix M: $CS_i = (Hash(M), i)$. This signature will be used in the TorCoin algorithm to prove that TorCoins are minted only by circuits that have been assigned by consensus groups.

Security Considerations

Anonymity The TorPath protocol guarantees that no single server knows the entire circuit of any client. If there are malicious colluding clients or relays, they will be able to shrink the anonymity set to the set of honest relays and clients in the consensus group. Groups have various sizes, allowing clients to choose the anonymity threshold they require.

Random Group Selection The TorPath protocol increases robustness of TorCoin to attackers. Its random group selection system prevents attackers from deterministically placing themselves in a group. We assume that at least one of the assignment servers in the group is honest. In this case, the group as a whole is able to retain privacy and anonymity. We could make the system even

more secure by randomizing group assignment, instead of just taking temporal locality to be the only criterion.

Same relay in multiple positions It is possible that through a Neff shuffle, the same relay gets assigned to a circuit in multiple positions (e.g., the same physical relay could be both the entry and middle relays.) With a reasonable number of participating relays, however, it should be extremely unlikely that one relay gets assigned to *all three* positions on the same circuit. In any case, the risk of accidental relay duplication on one path should not be substantially greater than the risk Tor users already face of randomly choosing multiple relays owned by the same operator.

3.4 TorCoin Mining

In contrast with Bitcoin's reliance on proof of computational effort, mining TorCoin requires proof of Tor bandwidth transfer. We introduce a novel method to prove end-to-end bandwidth transfer across a Tor circuit, in which all four participants of a circuit assigned by TorPath may collectively mine a limited number of TorCoins, incrementally, based on the amount of end-to-end goodput they observe on the circuit.

Proof of Bandwidth

1. Each client and relay creates a temporary key R and its hash $R'_* = Hash(R_*)$.
2. Every m Tor packets, the client sends a tuple (coin#, R'_C), where coin# is the number of Tor packets sent in this circuit.
3. Each relay adds its own temporary hash to the tuple and sends it to the next relay in the circuit.
4. The exit relay adds its own hash to the tuple to create the coin blob $B = (coin\#, R'_C, R'_E, R'_M, R'_X)$
5. The exit relay then signs the blob B to create S^B_X and sends the tuple (B, S^B_X, R_X) to the middle relay.
6. Each of the relays, i, signs the blob B to create S^B_i and adds their signature and their temporary key to the tuple, and forwards it to the previous component in the circuit.
7. The client receives the tuple $candidate = (B, S^B_X, S^B_M, S^B_E, R_X, R_M, R_E, R_C)$ and verifies its correctness, thus proving bandwidth transfer.
8. To check if a TorCoin has been minted, the client checks if the lower-order bits of $Hash(CS_i, candidate) == 0$. If so, the client adds the tuple $(CS_i, candidate)$ to the blockchain to mint a "TorCoin".
9. The client then pays each relay in the circuit $\frac{1}{3}$ of the TorCoin.

All the information necessary for verifying proof-of-bandwidth is in the blockchain. Any interested party can then verify that the route signature is authentic and refers to the correct group by referring to the public log. They can also verify that the blob B was signed by the correct relays by verifying the signatures, as well as verify that the temporary keys R actually hash to the claimed values R'.

Security Considerations

Enforcing packet rate All honest relays and clients enforce the TorCoin packet rate m. Any relays or clients that deviate from this are reported to the assignment servers and the circuit is terminated.

Enforcement of TorPath circuits Relays know their neighbours' IP addresses and will refuse connections from any other IP address. Even if malicious relays connect to each other, they will not be able to sign each other's TorCoins since they do not have the circuit signature.

Compromised circuits Any system of colluding clients and attackers needs to control all four components of a route to mint a TorCoin fraudulently. Even if an adversary controls up to half the network, there is a probability of only $(\frac{1}{2})^4 = \frac{1}{16}$ that an adversary client gets a path of three colluding relays. In practice, gaining control of half of all Tor clients and relays would be hard for an adversary. To limit the impact of these malicious circuits, the network imposes a limit on the number of coins that can be mined by a particular circuit. This coin number is included in the blockchain, so it is easily verified. The impact of these compromised circuits can be further reduced by ensuring that consensus groups expire at frequent intervals, requiring clients to change their circuits.

Drawbacks The TorPath network is not backwards compatible with the existing Tor network, due to the fundamental differences of route assignment and access control, which are missing in Tor, but are necessary for the TorPath and TorCoin schemes to work.

However, any given physical relay or server can run both services at the same time. TorCoins will, of course, be generated only for measured TorCoin traffic.

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 [15] 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 1752 bytes. This can be broken down into:

- The first packet from client to entry relay: 34 bytes.
- Packet forwarded from entry to middle relay: 66 bytes.
- Packet forwarded from middle to exit relay: 98 bytes each.
- Packet from from exit to middle relay: 324 bytes.
- Packet from from middle to entry relay: 518 bytes.
- Packet from from entry relay to client: 712 bytes
- Total: 1752 bytes

Each round of TorCoin generation and verification happens only after m 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 $m \geq 10$, the TorCoin protocol overhead is around 5%. The value of m 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.

It is possible for the system to tune the value of m to decrease 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 Future Work and Conclusions

We have introduced a novel way to securely and anonymously assign paths to Tor clients. This scheme was motivated by the need to verifiably sign TorCoins, a bandwidth measurement scheme that incentivises bandwidth throughput over the Tor network. The TorCoin protocol is robust to malicious relays and clients colluding to mint TorCoins without transferring bandwidth.

Further work could investigate the use of cryptographic accumulators for circuit signatures, which could reduce the packet overhead during bandwidth-measurement and make the protocol more efficient.

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. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system. Consulted **1** (2008) 2012

9. Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z., Weiss, W.: An Architecture for Differentiated Services (1998)
10. Dovrolis, C., Ramanathan, P.: A case for relative differentiated services and the proportional differentiation model. Network, IEEE **13**(5) (1999) 26–34
11. Snader, R., Borisov, N.: Eigenspeed: secure peer-to-peer bandwidth evaluation. In: IPTPS. (2009) 9
12. Snader, R., Borisov, N.: A tune-up for tor: Improving security and performance in the tor network. In: NDSS. Volume 8. (2008) 127
13. Karame, G., Androulaki, E., Capkun, S.: Two bitcoins at the price of one? double-spending attacks on fast payments in bitcoin. IACR Cryptology ePrint Archive **2012** (2012) 248
14. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM conference on Computer and Communications Security, ACM (2001) 116–125
15. : Twisted Framework. https://twistedmatrix.com/trac/

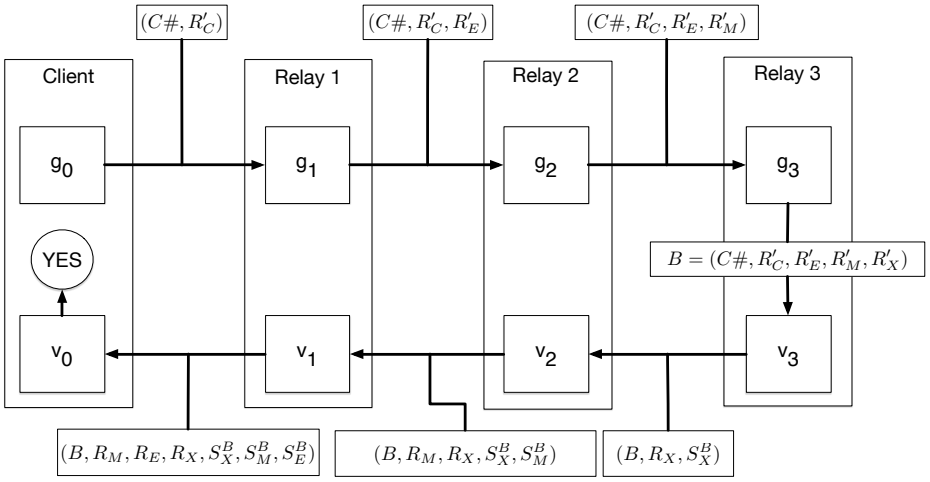

Fig. 4. TorCoin Proof of Bandwidth algorithm. In the upper part of the cycle, the relays add their hashes to the tuple. In the lower part, they add their temporary keys and sign the tuples.

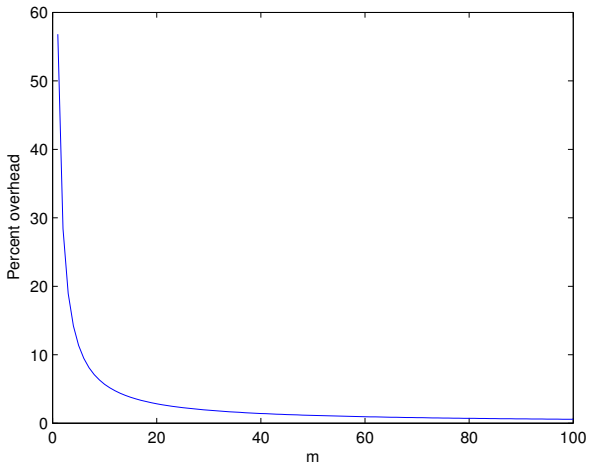

Fig. 5. TorCoin packet overhead