AlexTorPathCoin - A Cryptocurrency with New Money Generation from Multi-Party Onion Routing.

Alex Lawson

November 2021

Abstract

We provide a 'Proof-of-Bandwidth' alt-coin protocol that incentivizes participants to participate in Onion-Routing, like that of the Tor Browser. New currency is generated to a reward users for verified message routings. A path determination protocol is included that protects the anonymity of clients, while restricting adversarial clients from generating currency without routing, and requiring no central or trusted authority. Additionally, it allows for substantial decreases in total path latency when compared with randomized router assignment protocols like TorPath/TorCoin.

1 Introduction and Background

In the standard Bitcoin protocol a 'block', including a list of transactions and the creation of a new coin for the miner, is published when hashing the whole block with an arbitrary 'proof of work' string produces a special output. Miners collect transaction information, and spend computational work generating strings to test for a special hash. This leads to a system where participants preferentially treat the longest block-chain, as the block-chain with the most computing power behind it.[1] Miners do the job of listening in and compiling transactions, but the proof of work doesn't prove particularly useful work.

Instead, we introduce a protocol where 'proof of work' is replaced with a 'proof of service': the 'service,' required to generate valid blocks, is working as a node on an 'onion routing network'.

In an onion routing network, like the Tor browser, Alice wants to send a message to Bob satisfying two criteria: eavesdroppers cannot know the message, and they cannot know that a message was sent from her to Bob. The first is addressed in Tor with public key cryptography. To have the second criterion satisfied, the network has a number of available intermediate routers, from which Alice chooses a sequence: $r_1, r_2, ...r_n$. Each router has an available public key $K_1, K_2, ...K_n$, and private key $k_1, k_2...k_{n-1}$. So that no router (but the last)

knows her message, Alice wraps her message message m in many 'layers' of encryption: $Enc(K_1, ...Enc(K_{n-2}, Enc(K_{n-1}, Enc(K_n, m))))$. These layers give the name 'Onion Routing'. The intermediate parties decrypt the code using their secret keys $k_1, k_2...k_n$, or 'peel off' one layer at a time. Since what remains in still encrypted, the intermediate nodes or 'routers' do not learn the message m.[2]

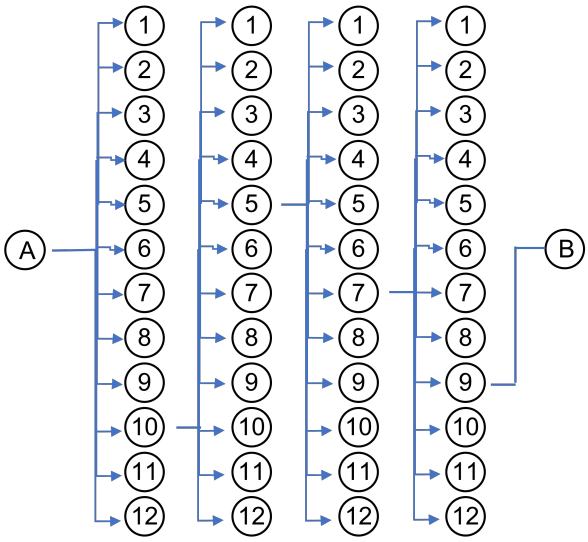

Figure 1: In standard Tor, Alice has full choice over which routers she uses in her routing, allowing her to optimize for latency and bandwidth.

The Tor protocol, which intends to be a free public service, requires volunteers. Due to this, there is a scarcity of routers. We present an alternate system where there is a financial incentive to routing encrypted messages: the generation of new units of the currency.

In this protocol, called AlexTorPathCoin (ATPC) rather than generating random numbers to get a special hash for the block, a block is officiated, and the publisher creates new currency when the they hit a certain number of "verified routings", which would be published on the block.

1.1 TorPath/TorCoin

A similar protocol to the one presented here is given in Ghosh et al.[3]. It introduces TorPath, a system for establishing the routing paths that Alice will use, and in less detail, complementary TorCoin, an alt-coin generated to reward routers.

As relays are financially rewarded, TorPath and ATPC need to restrict Alice's ability to performing a 'fake routing' with colluding relays of her choice,

where accreditation for routing is done without actually routing. In TorPath, trusted assignment servers are used to establish relay paths. Alice and other clients request a relay path, and those paths are (verifiably) shuffled to give Alice's true message path, which is practically random. Ghosh et al. focuses on the protocol for establishing these paths without any participant (including Alice) learning which routers are used in a path (aside from those with which it communicates directly).

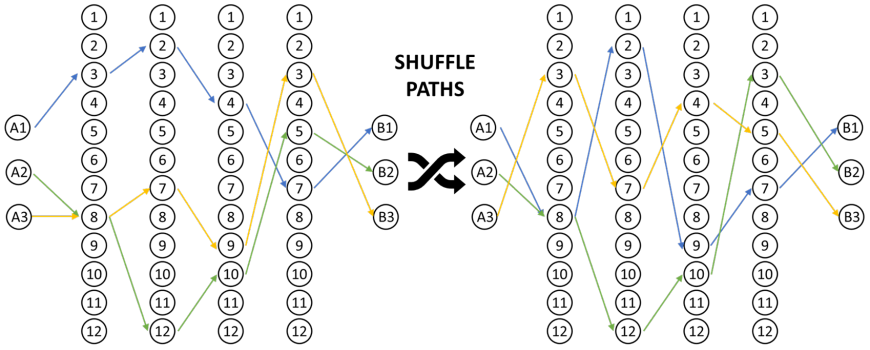

Figure 2: In TorPath, trusted assignment servers shuffle desired paths to make true paths, where clients cannot choose paths and collude.

TorCoin is not provided in much detail, and the protocol for paying out relays with new currency without revealing the path publicly is not addressed.

2 Protocol for AlexTorPathCoin

In ATPC, nodes for onion routing are also miners, marked by $n_i \forall i \in I$, where I indexes the miner/nodes. 'Nodes,' 'Relays,' 'Routers' and 'Miners' are used interchangeably. 'Publish' will be taken to mean 'send to all nodes', whereby all nodes will incorporate the information into their next published blocks. IP_i will be used to denote the IP address of n_i, i.e. if a node reads n_i, they know where to send the message. For simplicity, we will assume all possible messages are of the same size, envisioning parallel protocols for different message sizes.

2.1 Preparation

1. Each Router/Miner generates a public key, secret key pair K_i, k_i, (via ElGamal Key Generation for example).
2. Clients (Alice) and routers inform miners/routers that they would like to participate in Onion Routing. The Block that publishes this information, includes a list of these N participants' IP address and public key (IP_i, K_i), numerically ordered by IP_i. The place in that numerical order is denoted ip_i.

2.2 Onion Routing

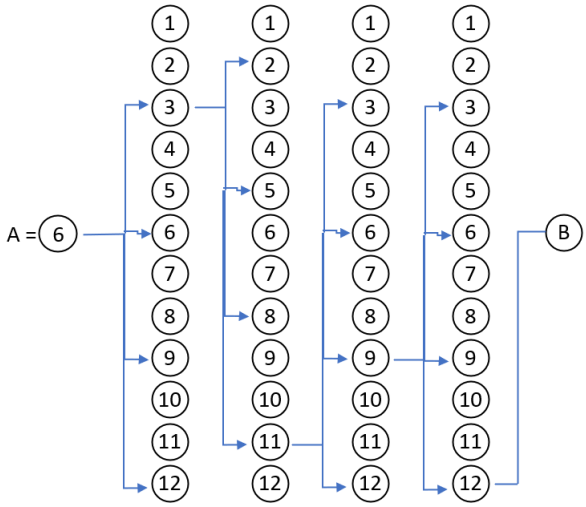

Figure 3: In ATPC, only a select number o relays are allowed as the next step in a routing path. This affords Alice limited choice in her path.

1. Alice chooses routers to send her message. At each step/layer, Alice has o of N options for next routers. For simplicity we assume o divides N, but this can generalize easily. These nodes need to be chosen is such a way that Alice cannot have *significant* personal preference for the nodes that route her message, and it is verifiable that she chose according to a set protocol. However, no one should be able to reconstruct the nodes Alice chooses with only public information.

   (a) List of those willing to route messages at given time is published in $Block_j$: $\{(ip_i, IP_i, K_i) \forall i \in [1, ...N]\}$.

   (b) Alice begins with her ip_{Alice}. At each step she chooses a next router via $ip_{i+1} = Hash(Hash(Block_j), ip_i) \mod N/o$. This was she has o options at each step. Alice does this for L total routers in the routing chain. This is illustrated in Figure 3

2. Alice Encodes her message m, via

$$c = Enc(K_1, IP_1|IP_2|Enc(K_2, IP_2|IP_3|Enc(\\ ...Enc(K_{L-2}, IP_{L-2}|IP_{L-1}|Enc(\\ K_{L-1}, IP_{L-1}|IP_l|Enc(K_L, IP_L|IP_B|m))))))$$

More cleanly Alice sends c_1 to n_1, where

$$c_i = Enc(K_i, IP_i|IP_{i+1}|c_{i+1})$$
$$c_L = Enc(K_L, IP_L|IP_B|m))$$

In this formulation router L sends m to Bob. We assume m is already encrypted by Alice and decryptable by Bob.

3. Each node n_i performs these operations:
   (a) Receive c_i. Let $x|y|z = Dec(k_i, c_i)$.
   (b) If $x \neq IP_i$: Conclude "unsuccessful routing".
   (c) Otherwise: Note n_{i-1} succeeded in routing a message of size $size(z)$. n_i concludes that n_{i-1} did a "good routing". Send $z = c_{i+1}$ to $y = IP_{i+1}$.

   We'll note that each router has no knowledge of other routers being used except those it sends to and receives from. Except the first and last router, they have no knowledge of Alice and Bob.

4. Bob receives m. Bob notes that n_L did a good routing.

2.3 Acknowledgement and New Money Generation

At the end of the message sending protocol there is a step for each node to 'acknowledge' or 'accredit' the node that send a message to them. This is the source of new money generation.

1. Each node (including Bob) keeps a record of nodes who did a "good routing" to them, and participates in the routing protocol until it acknowledges t different senders for $Block_i$. t should be $\leq o$, though an exact value is not discussed here.

2. Each node creates the message "$\{n_i\}$ did a good job routing a message using $Block_i$, where the set $\{n_i\}$ is the set of t nodes that this particular node noted for sending good messages. Each node signs this acknowledgement with their own verifiable digital signature, (via RSA) and sends this acknowledgment message to the other node/miners.[1]

3. All nodes listen for t-acknowledgement messages from other nodes. Once a given node n_i has work acknowledged in t messages, and has acknowledged the t who sent to them, they have done the necessary work, and are ready to publish a Block.

[1]It is not necessarily true that each node will receive messages from t different senders. For example, it is possible but unlikely that no ip_j in the list has $Hash(ip_j) = ip_k \mod N/o$, though in this case, the node ip_k would not participate in the routing at all. Still, it is possible that less than t routers end up routing to ip_k. In this case, publishing a list of less than t distinct routers would partially compromise the anonymity of clients. Therefore simply suggest the router doesn't publish any accreditation at all.

4. n_i publishes a Block, which includes:
   (a) Hash(Previous Block)
   (b) List of transactions sent to miner (as in typical cryptocurrency)
   (c) List of those willing to participate in next onion routing, with ordered IPs and public keys.
   (d) Acknowledgements of t different 'good routings' sent by other miners.
   (e) Statement that miner n_i receives 1 coin.

3 Discussion

3.1 Relay Accountability

Relays are rewarded if other relays accredit them with sending along the message. In order to receive currency, a router has accredit and be accredited, so they are incentivized to accredit. However, a dishonest router may choose to accredit the wrong IP addresses, given that they hash to the dishonest routers IP. This router would have the extra work of hashing candidate ip addresses to find one that hashes to theirs, so it would take more work than simply accrediting honestly.

3.2 Hiding Alice's Relays

By the accreditation messages published, one can partially reconstruct the path that Alice's message took. Assuming all relays publish accreditation, as Alice is accredited, one can show there are t^L possible paths for her message to take. t and L can be easily chosen such that t^L is greater than the total number of relays participating, in which case no new information about Alice's message is revealed.

TorPath does not address this issue, as the system for payouts is not the focus of the paper. Standard Tor does not need to deal with this issue, as relays are volunteers and paths are not published in any way.

3.3 Fake Message Sending

Since each relay in the chain is rewarded for relaying the message, we would like to limit the amount of preference toward certain relays that the sender can have. We imagine Alice is dishonest, and wishes to send a message to only routers she controls. If a real message of appropriate size is actually sent through, Alice may as well have simply volunteered to route messages in the traditional way. So in this model, Alice doesn't route a message at all. Instead she tries to find a path of 'fake accreditation', where each router in the path accredits the one before it, but sends no message.

Since the Block used for hashing through IP addresses is already established with the list of IPs, Alice cannot acquire IP addresses in order to artificially

construct a valid path. Instead analysis is presented to look at the probability she can construct a 'fully cheating path', one where each node along the path is colluding with her.

It should be noted now that even this would not give Alice significant advantage. Since each router is only rewarded if it is accredited by t different routers, Alice's routers must still participate with multiple different routers to be rewarded, and these routers will be checking that she routes full sized messages, earmarked for them. As much of this work was dedicated toward fully colluding paths, that analysis is still presented.

We choose not to dig too far into 'partially colluding paths.' In a 'partially colluding path', Alice must send and receive fully sized and encrypted messages with the non-colluding routers, but doesn't waste bandwidth on her own nodes. There is still some advantage in this, though Alice will have to cooperate in some capacity.

In the case of a fully colluding path, assuming that Alice has controls a fraction $a = \text{Num Colluding}/N$, of routers, her probability of creating a fully colluding path of length L is given below:

$$P_L = \sum_{k=1}^{o} \frac{o!}{(o-k)!k!} (aP_{L-1})^k ((1-a)P_{L-1})^{(o-k)}$$

For a path length $L = 10$, the (log) probability Alice constructs a fully colluding path is shown as a function of a in Figure 4.

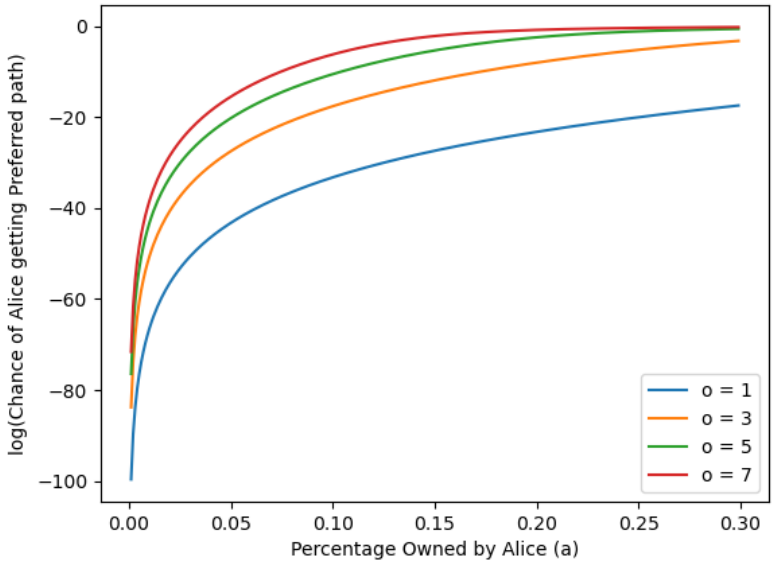

Figure 4: $\log_2$(Probability that Alice constructs a fully colluding path) as a function of colluding portion a, for different option numbers o.

3.4 Minimizing Latency

Though Alice's ability to choose between possible paths increases the likelihood of a 'fake routing' exploiting the currency generation system, there are benefits to the choosing system when Alice is truthful.

We will focus on comparing the expected latency of this protocol to that of TorCoin and standard Tor. A similar analysis for Bandwidth could be performed: this protocol has been formulated in terms of a one-time message transfer, though it could be used to set up a continuous message transfer path. Given some initial probability distribution of routing latencies: $\rho(\tau)$ and probability the latency is less than τ: $P(\tau) = \int_0^\tau \rho(\tau')d\tau'$, Alice can chose routers to minimize the total latency of her message path, which we will say is the sum of latencies for each relay.

We'd like to see the expected latency of a 'minimal latency' path that Alice can construct. This is inherently a traveling salesman problem: Alice may need to choose a higher latency relay if the relays following it lead to a smaller total latency. Seeing that we only intend to compare the three protocols, we will not implement a full traveling salesmen analysis. We simply ask the expected value of the total latency if Alice runs a greedy algorithm for choosing relays: choosing the lowest latency at each step. This will serve as a decent proxy for the minimum latency path. Additionally, it will be a better representation of

the path latency Alice should expect, as possible paths exponentially increase with each layer, (o^L), and computing all possible paths for each hash may be computationally intractable.

For o different options at each layer, the expected minimum latency $\bar{\tau}_{\min}$ is given by:

$$\begin{aligned} P(\tau) &= \int_0^{\tau} \rho(\tau')d\tau' \\ P(\text{All } o \text{ greater than } \tau) &= (1-P(\tau))^o \\ \text{Probability density}(\text{Min} = \tau) &= -\frac{d}{d\tau}(1-P(\tau))^o \\ &= o\rho(\tau)\big(1-P(\tau)\big)^{(o-1)} \\ \bar{\tau}_{\min} &= \int_0^{\infty} o\tau\rho(\tau)\big(1-P(\tau)\big)^{(o-1)}d\tau \end{aligned}$$

$$\bar{\tau}_{\min,\mathbf{tot}} = \int_0^{\infty} oL\tau\rho(\tau)\big(1-P(\tau)\big)^{(o-1)}d\tau \tag{1}$$

We choose a Gamma Distribution, $\rho(\tau) = \frac{1}{\Gamma(k)\theta^k}x^{(k-1)}e^{\frac{-x}{\theta}}$ with mean $\bar{\tau} = \theta k$ and standard deviation $\Delta\tau = \theta\sqrt{k}$, to represent our distribution of latencies. Unlike a normal distribution, it has the property that for $k > 1$, the $p(\tau)$ approaches zero when τ approaches zero, as we cannot have negative latencies.

For TorPath/TorCoin, Alice has no choice over the path, and is subject to the latency of the participating relays: $\bar{\tau}_{\min,\mathbf{tot}} = L\bar{\tau}$.

For standard Tor, Alice has full choice over the relays she uses, which will translate to a better total latency. In this case, we look at the total of the top L of N router latencies, N is the total number of routers. This was calculated by numerically sampling from the gamma distribution to choose the minimum latency L routers of the N. The expected total latencies found for each of these methods are presented in Table 1.

$\bar{\tau}$	$\Delta\tau$	Tor, $N = 1000$	TorPath	ATPC $o = 5$	ATPC $o = 10$	ATPC $o = 20$
4	1	19	40	29	26	24
10	3	38	100	68	59	53
4	3	1.52	40	13	8.3	5.4

Table 1: Expected Total Latencies $\bar{\tau}_{\min,\mathbf{tot}}$ for Standard Tor Protocol, TorPath, and AlexTorPathCoin with path length $L = 10$ and option number $o = 5, 10, 20$. $N = 1000$ participants

As can be seen in Table 1, the choice over routers has a substantial increase over the random allotment of routers in TorPath. As o is increased, it becomes closer to the expected latency with Tor, where we assume Alice uses only the best L routers. For the case of $L = 10, \bar{\tau} = 10, \Delta\tau = 3$, expected latency is graphed as a function of option number in Figure 5.

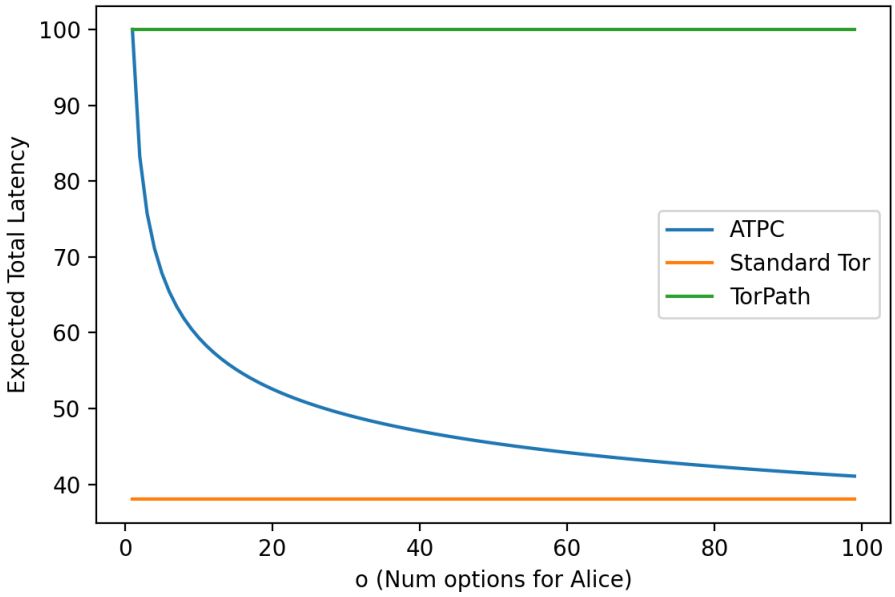

Figure 5: Expected Total Latency $\bar{\tau}_{\min,\mathbf{tot}}$ for AlexTorPathCoin as a function of option number o. Compared with Standard Tor Protocol and TorPath for path length $L = 10$, $N = 1000$ participants, average latency $\bar{\tau} = 10$, and latency standard deviation $\Delta\tau = 3$.

It can be seen that only a small number of options (5) at each step are necessary to half the difference in expected latency between the best routers, and randomly picked routers.

4 Conclusions

Despite some promising results, there are significant holes in this protocol. Participants are required to accredit t different routers at the same time, as they would compromise path information otherwise. These routers may chose publish unwillingness to receive messages from certain routers they have received from before. This lack of participation could increase the expected latency of clients. One possible remedy, is to payout based on the total message size relayed, allowing for multiple messages through one channel, though still requiring t different routers.

Additionally, as no work is required to accredit other routers, unlike with Bitcoin, there is a risk of an inflation explosion, with parties altruistically accrediting each other to acquire alt-coins without any routing. It is possible participants would not do this, noting that it devalues the currency, but they could preferentially accredit and not share the inflated wealth. There is no protocol in place for determining how much data needs to be carried through to

earn one coin. This should adjust as the currency develops, though suggested mechanisms are not addressed here.

More generally, with the current abundance of cryptocurrencies and alt-coins, it is not certain why ATPC would have any value. It could be possible to build in some system where only ATPC could be spent, like paying for preferential routing speeds, though the distributed nature of this protocol makes thsi hard to enforce.

We have shown a 'Proof-of-Bandwidth' protocol that incentives participants to participate in onion routing through alt-coin generation as a reward for verified transactions. This protocol protects the anonymity of clients, while restricting adversarial clients from generating currency without routing (given proper choice of o, L). Additionally, it allows for substantial decreases in total path latency when compared with random router assignment protocols like TorPath, putting it closer in expected latency to the router choice inherent to Tor for modest o.

References

[1] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Web document., 2008.

[2] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium (USENIX Security 04)*, San Diego, CA, August 2004. USENIX Association.

[3] Mainak Ghosh, Miles Richardson, Bryan Ford, and Rob Jansen. A torpath to torcoin: Proof-of-bandwidth altcoins for compensating relays. 2014.