Pzdid: A Decentralized Network Routing Market

Abstract

We present Pzdid: a decentralized market for anonymous communication and virtual private networking. Extant privacy solutions are either opaque commercial services with concomitant centralization risks or free peer to peer networks which lack the proper aligned incentives for service quality and economic security at scale. Pzdid is a bandwidth market where node providers stake tokens to advertise their services using the Ethereum blockchain. Clients construct single or multi-hop onion routed circuits by selecting nodes randomly weighted on stake and filtered on secondary criteria (price, location, etc.). Staking aligns incentives against operator malfeasance and linear stake weighting in particular neutralizes sybil attacks. Pzdid uses a probabilistic payment system which scales to millions of transactions per second, enabling a highly liquid bandwidth market without a trusted central party. Payments at packet scale allow high frequency trustless interactions by reducing the implicit floated balances between transactors to miniscule levels.

1. Introduction

The Internet, once a free and open frontier, is today increasingly fractured, surveilled, and censored. As governments and corporations have become ever more effective at monitoring, inspecting, and blocking connections, demand for privacy and anonymity tools such as VPNs (Virtual Private Networks) has grown mainstream. While VPNs work well enough for most use cases, they suffer from inherent weaknesses in the centralized trust based model. Users have little assurance that their VPN provider is not secretly logging and sharing data due to government coercion or the lure of additional revenue. The recurring payment and pricing models of VPNs create lock-in effects, preventing users from cheaply and rapidly switching between providers when one is blocked or slow. Current peer to peer systems such as Tor[1] or I2P[2] construct multi-hop circuits to hide route information from any single party. However these systems are free and thus suffer in terms of both performance and security. Performance and quality suffers because of poor incentives and the very limited supply of donated free bandwidth. Security likewise suffers from the low takeover cost for an attacker to provide a significant fraction of total network bandwidth.

What is needed is a peer to peer privacy network with proper economic incentives and nanopayments, allowing clients to construct single or multi-hop routes from a unified global pool of nodes from many distinct providers. An open market system can ensure that the supply of bandwidth, provided by profit motivated sellers, can scale elastically with growth in demand from users. The use of cryptocurrency contractual mechanisms can provide the necessary incentives against malicious behavior.

There are several core challenges driving our design: traffic analysis, sybil attacks, and the random selection problem. We describe each briefly before describing Pzdid itself in detail.

Traffic Analysis

It is quite difficult, in theory and practice, to send a message without leaking any information to parties other than the recipient. In A *mixing network,* first proposed by Chaum[3], messages are routed through a number of proxy nodes, randomly reordered at each step, and encrypted in layers like envelopes containing envelopes. *Onion routing*, a later development adopted by Tor[1], uses the same layered encryption concepts combined with unique random proxy node paths (circuits) for each persistent connection instead of a single shared circuit for greater scalability. *Traffic analysis* is still a potential problem[4], but can be overcome at a significant performance cost by bandwidth burning (padding) and/or random message delays. *Collusion* is another serious problem: if at least every other node in the circuit is cooperating they can infer the complete circuit.

Sybil Attacks

In any open network, agents can create many fake identities, presenting as a large number of independent nodes which in fact are all actually colluding. Retaining the property of openness while preventing a single attacker from overwhelming the system can be difficult. One solution to this problem is *proof-of-work*, originating in HashCash[4], later adopted by Bitcoin[5], and proposed as a sybil defense in the earlier Pzdid 0.9.2[6]. Proof-of-work requires each node to expend computational resources to prove their identity. Creating many fake identities would thus require a proportionally higher cost expenditure. *Proof-of-burn* is similar in effect but requires only proof of destruction of crypto-currency, which has the advantage that the value of the currency burnt is redistributed to the currency stakeholders rather than fully wasted. *Proof-of-stake* based cryptocurrencies require users to stake currency to receive block rewards and participate in the network. We use a *stake-weighting* system to defeat sybil attacks and align incentives, providing key economic security advantages.

Random Selection

To construct secure circuits with a low probability of collusion, we need to select randomly from relay nodes in a way which is immune to sybil attacks. We accomplish this with linear stake-weighted random selection, which is *Sybil-Orthogonal* : an attacker gains no advantage by dividing their stake into multiple identities. This selection scheme also provides a simple effective means of load balancing, and has subtle additional benefits even in the case of a minimal one-hop circuit (where collusion is less relevant). Implementing a global random selection policy requires that clients have available a global list of node metadata. The earlier Pzdid 0.9.2[6] proposed a custom Chord[7] based DHT (Distributed Hash Table) for this purpose. For simplicity we now use the Ethereum blockchain[8] (and it's underlying DHT) directly to provide the global node registry.

Overview

Pzdid is a decentralized platform enabling clients to compose high performance onion routed circuits with a variety of potential uses, powered by a new stochastic nanopayment system to fund such circuits. Bandwidth providers running Pzdid server software obtain and then stake Pzdid Tokens ("OXT", an ERC20[1] compatible crypto-currency) in an Ethereum directory smart contract to receive traffic and revenue in relative proportion to their stake deposit size. Clients find nodes through stake weighted random selection, which we have implemented as a smart contract function, using a tree data structure. Clients then pay nodes using probabilistic nanopayments sent as frequently as once per second. Multi-hop circuits can use an account per hop or indirect onion payment forwarding to reduce information leakage from the payments themselves. Circuits can fail for technical or economic reasons (ie when the circuit specific cost of a client's traffic exceeds their current budget), and are simply resampled on failure. The core mechanisms of our design are surprisingly simple, but naturally the devil is in the details.

2. Background

Privacy has long been a concern in networking, especially as ever more of our information moves online and more vulnerabilities are exposed each day.

Many of our foundational computer networking protocols[9] and practices arose in an academic or hobbyist culture of high trust[2] between 1961 and 1989[3] and are still used in modern phones, laptops, and desktops. All of them were fundamentally unhardened and blind to economics. The default operations are like a mail system full of machine typed postcards, lacking verification and subject to undetectable modification or replacement while in flight[4].

Internet Service Providers (ISPs) tend to be utility companies, which are notorious for cooperating with[5] (or being run by) authoritarian regimes[10] while manipulating services to harm users while improving their bottom line[11]. While ISPs tend not to utterly destroy the value of their data transmission services (with some exceptions[6]), academics certainly didn't think of their original task as designing protocols that minimized the degree to which ISPs can corrupt voluntary bilateral client/server relationships enabled by the existence of private data transmission pipe monopolies.

Even in non-authoritarian countries, cable companies, telephone companies, or specialized firms have begun to lobby representative governments to legalize commercial espionage[12] and explicitly violate the original norms around forwarding all packets[13]. Facebook's popularity has declined dramatically since 2014[14] (and in 2019 ranked 94th out of the top 100 visible organizations, slightly ahead of Trump Org, and the U.S. Government itself)[7]. However, users can simply stop visiting Facebook, and have begun to do so[15]. ISPs, by contrast, serve low viscosity markets, and 60 million Amercians are facing a literal broadband monopoly[16].

While attempts at hardening popular protocols have been made, there are few (arguably zero) generically fully safe fire-and-forget protocols. For example, SSH is relatively secure and widely used[17], but traffic analysis attacks were identified in 2003[18] and as of 2019 patching of the issue in real code is quite haphazard[8].

Unhardened protocols sent through untrusted ISP routers isn't a pressing concern for most users, but many people do access the internet through coffee shops, airport, or hotel WiFi. Spying, service degradation, and price gouging is somewhat common in all these situations, because these situations recreate the ISP-like incentives in small. When free WiFi implementations are sometimes attempted, the reduced technical budget can lead to buggy configurations which accidentally let users spy on each other. In public perception, all of these challenges and more blur together into a vague sense that the internet in general, and especially the Internet accessed via wifi, is full of spying in a confusing and potentially dangerous way.

In the corporate world, Virtual Private Network (VPN) technology began seeing mass adoption initially as a way to allow employees (especially employees who travel or telecommute) to create an encrypted tunnel from a wider (default untrusted) networking context back into a secure work intranet. This setup was called a "VPN" because the

tunneling software enabled people to “Virtually” be “inside” their secure “Private Network”. It does not fully solve the problem of protocol hardening (the shape and timing of the traffic tends not to be protected) but sending a mixture of hardened and unhardened protocols through such a tunnel at least protects against injection attacks, and some kinds of inference attacks.

The rise of VPN services in corporate environments enabled basically the same technology to be repurposed (using similar tunneling concepts) and offered to consumer markets. In this new ecosystem there is no employer to fulfill the role of a local trusted authority, leading to various attempts by technologists, entrepreneurs, and researchers to explore various solutions for more trustworthy secure networks. Consumer VPNs occupy a niche along this spectrum of possible solutions, Tor occupies another, and attempts to improve on Tor have typically foundered on the challenges introduced by incentives and payments (or lack thereof).

2.1 Consumer VPNs

A consumer VPN company cuts into the job of the ISP for the user. Whereas before the ISP had two jobs: (1) installing wires and (2) not spying on the unhardened data in those wires, now the ISP only has the first job (which they retain because they have a monopoly on the wires that go into the user’s home). The second job is done partly by the VPN tunneling software (encrypting the data) and partly by the VPN company: decrypting the data into less hardened streams and forwarding different substreams to different parts of the wider internet.

These services can protect the user’s traffic against much of the hazards of untrusted WiFi scenarios such as at coffee shops, hotels, airports, etc. They have also become popular for a variety of other use cases where customers desire to hide their IP address from websites and or hide their traffic from their ISP.

When the VPN is active, the VPN effectively becomes the user’s new ISP from the perspective of many privacy and trust models. However, this implies that any attack an ISP could previously perform can now easily be performed by the VPN provider. Like other centralized systems, VPNs are only as safe and trustworthy as their controlling corporate entity. Furthermore, their existing payment systems and business models require monthly or longer service commitments with steep price premiums for short contracts, leading to user lock in.

2.2 Tor, The Onion Router

Users seeking private internet connections have alternatives in the form of (mostly free) distributed systems. The most widely used such system is the Tor network[1]. The core concept behind Tor is to obfuscate traffic by sending packets through multiple randomly selected, statistically uncorrelated intermediate routers before reaching the final destination.

Unfortunately, distributed systems such as Tor come with their own host of problems. One of the primary issues is incentivization of good behavior for the network - such as increasing availability and bandwidth while decreasing latency. These problems can be overcome through economic incentivization mechanisms.

Incentivization in distributed systems began as a way to apply simple economic models to systems with the goal of driving good behavior. Early algorithms often used bartering such as tit-for-tat[19] for distributed resource allocation, modeling rewards and punishments with network primitives such as bandwidth and latency. While this approach has generally led to stable distributed systems, they often still suffer from seemingly intractable problems such as the free rider problem[20]. As decentralized systems began to develop, an explicit economic reward and punishment approach to peer-to-peer incentives began to emerge. These methods created an explicit measure of

economic utility for incentives, thereby allowing for fine-tuned approaches to driving good behavior and disincentivizing bad behavior.

2.3 Incentivized Tor

One of the first examples of incentivized peer-to-peer privacy networks appeared in incentivized Tor[21]. This first proposal by Ngan et. al. suggested a *tit-for-tat* strategy for allocating routing resources as an incentive mechanism. At its core, tit-for-tat offers a way to distribute resources towards a peer the same way it distributes resources towards you. If your peer acts uncooperatively, you do the same. If your peer acts cooperatively, you also do the same. In this fashion, the payoff matrix of the iterative decisions always results in a Nash equilibrium.

More recently, Androulakil et. al.[22] demonstrated how actual payments could be used to more directly encourage packet forwarding. At a high level, the design revolves around a hybrid of an anonymous payment scheme (used to pay the first node in a route), and chained micropayments for the rest of the circuit. This design implies a marketplace for packet forwarding. Ideally Tor users will tend to choose the peers that provide them with the best privacy, bandwidth, throughput, and latency, and in exchange for their services, will pay them using a digital currency. Note that now, the utility of sending a packet can be directly matched with monetary incentivization instead of being held against a difficult-to-quantify payoff matrix in the tit-for-tat model.

While the core idea of economic incentivization is incredibly powerful for driving desirable behavior in a peer-to-peer system, there are some inherent issues. Perhaps the biggest issue is the reliance on a central bank to mint tokens. This could be solved by using a decentralized cryptocurrency for payments, as we discuss later in this paper.

An alternative approach to the above model is incentivization through a proof-of-bandwidth model presented by Ghosh et. al. [23]. In this model, each peer in a circuit helps generate a new proof of minting that is initiated by the client after sufficient bandwidth is sent. This information is broadcast on chain, which then effectively pays all members in the circuit for forwarding the packets. While this protocol seems valid in theory, it relies on inflation to pay nodes, lacks market driven pricing and there are additional concerns about withholding attacks and other malicious behavior.

Ultimately, it seems difficult to introduce an efficient incentive mechanism into Tor that doesn’t expose more potential attacks.

2.4 Payment Channel Backed Routing

Payment channels can be used to route both information and money. A prominent example of this is the Interledger Protocol (ILP) introduced by Thomas and Schwartz[24]. The core idea behind the Atomic Swap method in ILP is to use Hash Time Lock Contracts (HTLCs) to set up a cryptographically verifiable micropayment channel that pays out tokens as a data packet is forwarded. Note that, unlike traditional payment channels, these micropayment channels settle on chain relatively infrequently, allowing for both amortized transaction fees and low latency. In the process, however, routes are not fully hidden from the network.

Khosla[25]introduces an onion-routing based plugin on top of ILP that allows for Tor-like functionality associated with these cryptographically verifiable micropayments. Their system uses an ILP payment circuit for every link in the multi-hop data circuit, significantly multiplying latencies, error probability, and complexity.

While payment channel backed routing methods have attracted significant attention as a promising layer two scaling solution for decentralized payments, their deployment and efficiency suffers from the need to conduct recursive routing. End users must deposit funds with one or more specific payment routers, requiring trust and introducing a form of counterparty risk. Routing a payment requires O(log N) steps and latency. Payments are not always routable depending on their size and the deposits available along key edges. Payment routes can completely fail causing long delays if critical edges fail to deliver. For these reasons payment channel networks are not widely adopted micropayment solutions in general and for onion routing in particular.

3. Goals and Limitations

Pzdid ’s mission is to empower people to understand and control the network activity of their own computers without fear of censorship, surveillance, or intermediation. In service of that mission we are building solutions for a wide audience using open source software to create a decentralized VPN market powered by probabilistic nanopayments over the Ethereum blockchain. Our design emphasizes scalability, decentralization, usability, simplicity and extensibility. Pzdid inherits some current limitations from Ethereum in terms of payment anonymity, scalability, and censorship resistance. Additionally our initial focus on affordable high bandwidth, low latency routing currently limits Pzdid ’s ability to defend against the most sophisticated theoretical traffic analysis attacks. These limitations are not obstacles for most of our main envisioned mass consumer use cases (section 6).

3.1 Goals

Scalability

The Pzdid nanopayment system scales to a few million users sending probabilistic transactions once per second on the current Ethereum blockchain (section 5.9), and could potentially scale to billions of transactions per second using sharding with Ethereum 2.0. The node selection process (section 4.3) allows clients to outsource node selection to server nodes in a trustless manner, allowing for lightweight Pzdid client implementations.

Decentralization

All components of our design, from nanopayments to node directory and discovery, are decentralized. The Ethereum blockchain is used to enforce a minimal set of contractual settlements required by a functional market. There is no special trusted party with outsize influence or control in Pzdid , assuming OXT stake is well distributed.

Usability

Usability is key to wide adoption, and the anonymity the system provides per user increases with the size of the user base. Our default client implementation ‘just works’ without requiring unnecessary user decisions for configuration or route management (although detailed configuration options are available for those who want them). The client also helps automate some tedious details such as budgeting and node selection. For most users, using Pzdid to protect their network connection is almost as simple as pressing a button.

Simplicity

The protocol is simple to ease comprehension, implementation, and security analysis. We use seller determined bandwidth prices and client price filters instead of more complex auction mechanisms. The stochastic payment protocol is also comparatively simple: the smart contract consists of around 200 lines of Solidity code.

Extensibility

Our core mechanisms are separable and orthogonal to the extent possible to permit easier future extension and replacement. The nanopayment protocol and smart contract do not directly interact with the other systems. The node directory likewise is isolated and separated from the node metadata registries and other components. Key system design hyperparameters, such as the withdrawal delay, were made contractual parameters where possible for ease of adaptation. The WebRTC based transport protocol is likewise orthogonal and extensible. The nanopayment system, while built for the Pzdid bandwidth market, is generic and has potential for broader uses.

3.2 Limitations

Pzdid is built on Ethereum, the world's leading blockchain in terms of smart contract capabilities, decentralization, community size and engagement. Thus we partake of any and all scaling and security issues inherent to Ethereum, but also can rely on the efforts of the extended Ethereum community to deal with any crises that may arise.

Network Dependence

Pzdid ’s economic security (section 4.4) is upper bounded by the economic security of Ethereum itself. An Adversary with the capability to destabilize or takedown the Ethereum network could naturally takedown Pzdid . (And furthermore, any successful shutdown attack on Ethereum would de facto also shutdown Pzdid , even if this was unintentional). A powerful Adversary could accomplish this by launching a sustained 51% attack, for example, perhaps amplified by DDOS and other attacks against key Ethereum nodes.

Pzdid server nodes also depend on the Ethereum network at the individual level, as they need reliable connections to Ethereum nodes to process winning nanopayment redemptions. Pzdid nodes are thus also individually vulnerable to Ethereum eclipse attacks. In practice commercial Ethereum node operators such as Alchemy or Infura help mitigate these risks.

User Scalability

The current Pzdid nanopayment system has an efficiency/variance tradeoff: larger face value tickets reduce the frequency of on-chain payments and transaction fees at the expense of variance. We expect users will have a limited tolerance for variance. Given these constraints and Ethereum’s current max transaction throughput of around a dozen transactions per second implies a scaling limit of a few million Pzdid users (section 5.9). Scaling beyond this user limit is possible with Ethereum 2.0 sharding[9].

Payment Anonymity

Rare winning nanopayment tickets are redeemed through on-chain Ethereum transactions. Thus Pzdid nanopayments are only pseudo-anonymous, and occasionally leak some information (section 5.8). Users desiring stronger anonymity will need to anonymize their OXT currency prior to loading it into a nanopayment account.

Public Node Directory

The Pzdid node directory is published on the Ethereum blockchain and thus is public to the world. Thus it is easy for a censoring Adversary to automatically block all the listed contact IP addresses of Pzdid nodes. Implications and possible workarounds such as using private IP addresses shared off chain are discussed in section 6.4.

9

Traffic Analysis

Our initial focus is on high bandwidth, low latency circuits at the expense of strong anonymity. This tradeoff is fundamental[26], but our design allow users the ability to trade off bandwidth efficiency for increased anonymity through bandwidth burning.

Traffic Obfuscation

Pzdid 's network layer is built on WebRTC which provides a certain initial capacity for obfuscation. However there is an ongoing research arms race between obfuscation and detection.[27] Sophisticated Adversaries can defeat most known obfuscation techniques; we leave stronger obfuscation plugins as future work (section 7).

4. Market Design

The Pzdid Market is a decentralized peer-to-peer (P2P) network which allows users running the Pzdid client to purchase bandwidth from one or more sellers running the Pzdid server in order to form a proxy circuit to a specific resource on the Internet (such as a website).

The main participant roles in the Pzdid Market are:

- A **user** running the Pzdid **client** who initiates proxy circuit connections
- (optionally) One or more **relay nodes** who forward encrypted traffic
- An **exit node** who provides the final connection to an external destination (e.g. website)
- A **bandwidth seller** who accepts nanopayments for traffic (either a relay or exit)

Bandwidth sellers register their nodes on the Ethereum blockchain and user clients select suitable nodes for routes all through calls to Ethereum smart contracts. Pzdid uses stake-weighting: sellers lock up OXT tokens to form stake deposits associated with their nodes in order to receive traffic in proportion to their relative stake.

4.1 Fundamental Operations

At a high level, the Pzdid Market provides the following key operations:

- A means for bandwidth sellers to register their nodes via staking
- A method for bandwidth sellers to register custom services and metadata
- A means for clients to query nodes for custom offered services and metadata
- A method for selecting a random node, with probability proportional to stake, such that the *Sybil Orthogonality* property holds (for node X, stake size S, and multiplier constant α):

$$P(select(X) \mid stake(X) = \alpha S) = \alpha P(select(X) \mid stake(X) = S)$$

Sybil Orthogonality requires a linear selection property which ensures that an attacker who splits up their resources into multiple sub-accounts does not gain an advantage in their selection probability and consequent expected connection requests per unit time; thus sybil attacks have no benefit. Given this linear weighted selection property,

then if there are any number of attackers with aggregate stake A out of S total system stake, then a randomly selected node is *not* an attacker with probability:

$$P(select(\neg Attacker)) = 1 - \frac{A}{S}$$

The use of stake-weighting allows the economic security of the Pzdid network to scale linearly with the size of the total deposited stake, which we can expect to be a sizeable fraction of the total OXT market cap (staking economics are analyzed in more detail in section 4.5 below). The stake-weighted selection process itself is implemented using an on-chain tree data structure described in section 4.3 below, which allows a client to outsource the selection of nodes to other nodes in a scalable trustless manner, avoiding the need for lightweight clients to ever download, store, or process the complete node directory.

4.2 Node Directory

The Pzdid node directory is a set of data structures stored on the Ethereum blockchain that allows clients to efficiently select bandwidth seller's nodes. Essentially it forms a simple Pzdid specific overlay over the Ethereum network. The node directory contract provides several main functions:

- **push** : a method to stake a variable *amount* of OXT tokens on a specific *stakee*, adding to an existing entry or creating a new stake deposit entry keyed on (staker, *stakee*). The push function also takes a *delay* parameter which will determine the subsequent withdrawal lockup period.
- **pull** : a method to initiate a pending withdrawal of a variable *amount* of OXT token from an existing deposit entry keyed on (staker, *stakee*).
- **take** : a method to finalize a pending withdrawal after the delay period, transferring the pulled funds to a regular liquid OXT ERC20 balance
- **scan** : a method to select a random node weighted by relative stake, given a random seed parameter

Node Metadata Registry

The Node metadata registry allows anyone to 'tag' nodes with metadata. Bandwidth sellers can use this to store custom metadata associated with their nodes on the blockchain and advertise services, constrained only by the gas cost of Ethereum transaction fees. The metadata registry is generic to enable a simple means for future custom extensions, allowing node operators to advertise new services which clients can then select for without code updates.

Node Directory Tree

To implement the scan function efficiently, we use an on-chain binary weighted tree data structure. Each node in the tree is a stake entry which stores a stakee, an amount, and a delay, in addition to the tree pointers and stake subtotals for the left and right subtrees. This structure effectively forms a prefix sum tree over all the stake deposits, allowing a simple descent decision at each node to find the subtree (or internode) containing a given random point; finding the exact node interval containing the given random point requires only a logarithmic number of steps.

Withdrawal Delay

The withdrawal delay is an important security restriction. It creates an obstacle for attackers attempting to acquire a large portion of Pzdid client connection requests. In particular we are concerned with preventing a *systemic takeover attack* where the attacker acquires a large fraction of the total deposit stake and then directs clients to

malicious servers which provide intentionally poor connections, log and report traffic, or attempt active connection attacks (e.g. SSL downgrade).

Similar to *proof-of-stake* (PoS) cryptocurrencies, our main defense against systemic takeover attacks is the high cost barrier to acquiring and locking up a significant fraction of the total OXT stake. Without a withdrawal delay this barrier becomes just one of access to sufficient liquidity with little actual net cost to an attack. A withdrawal delay creates a minimum interest or opportunity cost for a stake position. A successful attack will also disrupt the network and likely reduce the OXT token value. So a sufficiently long withdrawal delay is more likely to create an additional loss for the attacker when they finally end the attack and sell their large OXT position.

Although the underlying mechanisms are quite different, systemic attacks in Pzdid with short withdrawal delay are analogous to *rental attacks* in proof-of-work (PoW) blockchain systems. The rise of hashpower rental services such as Nicehash[10] has provided a large pool of hashpower liquidity which can be used to dramatically lower the cost of a 51% attack on a PoW system vs the alternative of purchasing the requisite hardware. Attackers have executed double-spending attacks on many smaller coins using rental hashpower, and even Ethereum Classic, a top 20 coin, was successfully attacked in early 2019[11].

The ideal withdrawal delay should be longer than the time we expect the market will need to detect and react to a systemic takeover attack. But longer withdrawal delays also impose an opportunity cost on honest bandwidth sellers who wish to reduce or exit their stake deposit position. The ideal tradeoff between these two constraints is difficult to estimate a priori, so we chose to make the withdrawal delay a flexible parameter. The client software then *filters* on withdrawal delay, ignoring stake deposits with delays below the client threshold. Our initial client software will accept withdrawal delays of 3 months or greater, but the flexible parameterization allows future client updates to change this parameter without the equivalent of a hard fork and associated coordination difficulties.

4.3 Node Selection

Clients select nodes for proxy circuits using a two step process of random relative stake-weighted linear selection followed by secondary constraint filtering. The first stage linear selection is performed by the scan function on the node directory tree. The client generates a random point locally and passes it in as the single argument to scan, which then descends down the node directory tree. The search terminates in the single unique leaf or internode whose stake segment intersects the chosen random point.

Using a smart contract to implement the main node scan function allows the selection process to be easily outsourced to nodes. A client can request one or more scan calls and have a remote node execute each scan locally and send back simple proofs of their correctness using the eth_getProof and eth_getStorageAt functions of the ethereum JSON RPC API[12]. This mechanism ensures that clients can provably trust that the node did not maliciously choose themself or an alias, and returned the same results as if the client had executed the function locally on their own full copy of the ethereum blockchain. Outsourcing the scan function allows for lightweight Pzdid client implementations.

After selecting one or more nodes based on linear relative stake weighting, the client can then optionally filter on a few additional criteria such as exit geolocation, latency/ping, node whitelists, or custom metadata tags.

Geolocation

A popular use case for VPNs today is bypassing geolocation based content filtering. Streaming services such as Netflix have country specific content licenses which are enforced by detecting a user's IP address. A VPN or exit server in the right location can thus allow access to otherwise blocked content.

It is difficult to prove that a particular IP address is actually within a specific location. Moreover the eventual server a client connects to may have a different IP address than that listed in the directory contract for legitimate reasons: a large bandwidth provider may bounce redirect incoming client connections to one of many proxy servers for load balancing. Due to these considerations, Pzdid clients interested in a particular exit geolocation can use the published node metadata to filter on the claimed geolocation, but ultimately must check whether the final exit connection actually is in the requested location. This check can be automated to some extent by the use of public IP address to geolocation databases.

Latency

We anticipate that in some use cases users will desire connections with lower latency than randomly chosen nodes. Clients can employ a guess and check strategy for latency similar to that used for geolocation. The claimed IP address can be checked against a public known database that maps IP addresses to locations to filter out distant servers. Ultimately the actual latency must be measured once a route is constructed. If the latency is higher than the target threshold, a new different route must be sampled. The lightweight nature of Pzdid routes and nanopayments allows for fast route setup and parallel route testing.

Price

As sellers set their own bandwidth prices, clients must be able to determine reasonable price levels to avoid egregious charges. The Pzdid client uses customizable budgeting algorithms to determine a current spending cap based on the user's balance and other parameters such as a target timespan representing how long the budget should last. For example, a user can load $50 worth of OXT into their nanopayment wallet and instruct the client to budget that money out over a year of bandwidth purchases. The client software then uses this budget to determine a limit on how much to pay over time. If the client pays less than what the server is charging for the bandwidth the client is using, the server will throttle their connection. If the throttled throughput is unacceptably low, the client will choose a new provider. Thus price forms an implicit filter, filtering out nodes with bandwidth prices that are incompatible with the client's current usage and budgetary spending rate.

Whitelists

The Pzdid client can use an on-chain curated list which filters the viable nodes to a custom subset. Initial releases of the official Pzdid client will use this feature to prevent certain kinds of attacks from malicious exit nodes (e.g. SSL downgrade attacks) by using a default exit node whitelist consisting of trusted VPN partners. Customized Pzdid clients can use their own whitelists, and eventually we expect well known third parties to emerge as whitelist curators. Whitelists are a simple means for the importation of external reputational trust to supplement the economic incentive based trust provided by staking.

Custom Metadata Tags

Bandwidth sellers can store arbitrary metadata tags associated with their nodes on the blockchain using a node metadata registry. In the future, sellers could use this to advertise new custom services, such as unshared IP addresses. Users can then have their client filter on the associated tag to find nodes claiming to offer that service.

Sellers guilty of false advertising (claiming services that they don't actually offer) run the risk of being delisted from popular whitelists.

4.4 Selection of Stake-weighting

Pzdid 0.9.2[6] presented a design based on proof-of-work medallions as the main anti-sybil mechanism, and explicitly argued against proof-of-stake. In this section we will analyze stake-weighting vs other alternatives, and why we moved to a stake-weighting approach similar to proof-of-stake.

Preliminaries: Attack Costs

Like Bitcoin, Ethereum, and most other decentralized systems, Pzdid is an open network built from open source software; anyone can download the Pzdid node software and run as many nodes as their resources permit. The viable defenses against systemic attacks in an open decentralized system are ultimately *economic*: a system is secure to the extent that the cost of an attack to an attacker outweighs the benefits to that attacker, or is too costly to execute regardless.

We can partition economic security into absolute and relative constraints. Relative economic security is the condition where an attack is *unprofitable*, regardless of the resources required. Absolute economic security is instead the security of a high cost barrier itself, which excludes attackers with insufficient resources. Bitcoin currently has absolute economic security measured in the tens of billions of dollars. A smaller new cryptocurrency may have far less absolute security, but could still rely on sufficient relative security to deter most realistic attackers.

Proof-of-Work

A proof-of-work system derives its security from the computational power that must be burnt to prove valid identity in the system. The Pzdid 0.9.2 design[6] used medallions that required *continuous* proof-of-work to maintain current active status, based on solving computational puzzles seeded on each new ethereum block. Thus the mechanics are quite similar to proof-of-work blockchain systems such as Bitcoin.

If we assume that the proof-of-work design is *not* ASIC-resistant so that specialized chips are dramatically more efficient than general chips, *and* we assume that no significant rental market exists for said chips, then a proof-of-work system's economic security constraint is approximately[28]:

$$N\,C > V_{sabotage} \qquad (3)$$

Here N indicates the total honest (non-attacker) hashpower, C is the total capital cost per unit hashpower, and $V_{sabotage}$ is the value the attacker derives from system sabotage. The lhs of equation 3 is the attack cost and also the absolute security barrier.

For bitcoin as of mid 2019, the value of NC is in the tens of billions of dollars. Bitcoin's proof-of-work specification is not ASIC resistant, and as a result, ASIC chips are dominant due to orders of magnitude higher efficiency than repurposable general purpose chips. Ethereum, on the other hand, intentionally designed an ASIC-resistant proof-of-work specification. As a result, ASICS have minimal advantage over general purpose graphics processing units (GPUs), which have dominated Ethereum mining. Being general purpose, there exists liquid rental markets for GPUs, and thus an attacker only needs to pay the rental cost of hashpower for the duration of the attack. If we ignore

the block rewards the attacker gains during the attack, the economic security constraint for a rental attack that takes t units of time with a rental cost of c per unit time per unit hashpower is approximately:

$$t \, \mathrm{N} \, c > \mathrm{V}_{\mathrm{sabotage}} \qquad (4)$$

As t, the time required for the attack, is generally orders of magnitude shorter than the depreciation timespan of hardware, the rental scenario leads to dramatically lower economic security. The proof-of-work medallion design in Pzdid 0.9.2[6] intentionally relied on equihash[], an ASIC-resistant scheme. This was somewhat necessary given the requirement that medallions must be generated by end users, many of which will have only cellphone level hardware. An ASIC-friendly proof-of-work algorithm would then give a huge relative advantage to an attacker with ASICS vs end users with cell phone CPUs. Unfortunately the use of an ASIC-resistant algorithm implies liquid rental market conditions and thus the lower security of equation #4 above.

The computation spent on proof-of-work puzzles is wasted, so it forms a kind of tax on the system relative to the net value of bandwidth the system provides. The revenue per unit time, P , then equals the cost of bandwidth, B , plus the implicit cost of compute required to maintain medallions:

$$\mathrm{P} = \mathrm{B} + \mathrm{N} \, c \qquad (5)$$

Economic considerations constrain Nc and B to be of similar order, as otherwise Pzdid would be too expensive for consumers versus alternatives. Substituting eq 5 into eq 4 we have the security condition:

$$t \, (\mathrm{P} - \mathrm{B}) > \mathrm{V}_{\mathrm{sabotage}} \qquad (6)$$

As a concrete example, consider a scenario where Pzdid has 1 million users who are each paying in total about $63 per year (wholesale bandwidth cost plus the implicit proof-of-work compute cost), and assume that proof-of-work overhead is roughly 50% of cost. The term P - B is thus only about $1 per second. With these parameters it would cost an attacker only about $3,600 worth of rented compute to capture about half of all Pzdid traffic for one hour, or about $86,400 worth of compute to capture about half of all Pzdid traffic for one day.

Stake-Weighting

In our current stake-weighting approach, bandwidth sellers stake OXT currency in time locked deposits to prove identity and receive traffic in proportion to relative stake deposit size. First we will assume that there is no market for borrowing OXT with sufficient available liquidity to be useful for an attack. To acquire control over 50% of Pzdid traffic, the attacker must acquire and stake an amount of OXT equal to the total not-attacker stake. A successful attack will lead to a drop in the exchange value of OXT; the main cost to the attack is the loss on the stake position. If S is the total honest (non-attacker) stake deposits, and $\mathrm{x_d}$ is the resulting (expected negative) percent change in the exchange value of OXT after the attack, then the relative security condition is just:

$$-x_{\mathrm{d}} \mathrm{S} > \mathrm{V}_{\mathrm{sabotage}} \qquad (7)$$

The attack cost and absolute security barrier is just S (size of stake deposits), as the attacker needs to spend capital of size S to execute the attack.

We can expect that bandwidth sellers will learn to increase or decrease their stake deposits in response to market conditions to optimize total profitability. The requirement that bandwidth sellers must lock up OXT currency to

receive traffic implies an implicit opportunity cost on their capital. In competitive equilibrium we can expect that the total gross revenue flowing to bandwidth sellers, R , will roughly equal their cost of bandwidth, B, plus the opportunity cost or interest rate per unit time, I_r, multiplied by the required stake capital:

$$R = B + I_r S \qquad (8)$$

The total stake S can then be rewritten in terms of the cost of bandwidth, the revenue flow, and the interest rates as:

$$S = (R - B) / I_r \qquad (9)$$

The opportunity cost of stake deposit capital is a form of overhead that has a similar role to the cost of burnt compute in the proof-of-work example. If we make the same assumption of an overhead of 50%, then the opportunity cost equals the cost of bandwidth. Using the same parameters from the earlier example, with 1 million users buying $63 worth of bandwidth per year, with 50% of that going to supplier bandwidth cost, and assuming an interest rate or opportunity cost of 10% per year leads to a total stake amount S of $315 million via eq. 9, which also is the absolute attack cost constraint from equation 7. This is more than three orders of magnitude larger than the attack cost using continuous proof-of-work medallions.

Now consider the scenario with a liquid market for OXT stake rental. We can first imagine a financial market where borrowers put up collateral in another currency, similar to a short position but without constraints on the use of funds. This type of rental market would not change the attack cost and absolute security constraint of S, but it would lead to different dynamics for the relative security constraint, as the attacker now avoids any loss from a drop in the value of OXT.

More useful to an attacker would be a market that rented out stake deposits directly, without collateral. As the deposits are illiquid, the renter can not spend them, but instead would have access to the full benefits of the stake deposit in terms of Pzdid node traffic. In this scenario the attack cost, relative and absolute security constraints are modified to a flow equation with only an interest cost:

$$t\, I_r S > V_{sabotage} \qquad (10)$$

In eq. 10 above, the attack cost is now just the interest on renting 50% of the pre-attack total stake (the size of the rest of the ‘honest’ stake S) for the duration of the attack *t*. Substituting the rhs of eq. 9 for S in eq. 10 leads back to the same eq. 6 earlier from the proof-of-work section:

$$S = (R - B) / I_r \qquad (9)$$
$$t\,(P - B) > V_{sabotage} \qquad (6)$$

So the worst case for stake-weighting where stake is fully rentable leads to a similar weakened security condition as proof-of-work where hashpower is fully rentable.

However, the withdrawal delay parameter puts a lower bound on the crucial attack time parameter *t*. Using the same parameters from earlier with 1 million users paying $63 a year and 50% overhead, a withdrawal delay of 3 months leads to an attack cost of about $7.9 million, which is still a few orders of magnitude larger than the attack cost for proof-of-work medallions.

We could suggest an even larger stake withdrawal delay, but it is unlikely that economic security increases monotonically with withdrawal delay. The withdrawal delay creates an additional opportunity cost for honest

participants exiting their stake position, and if that cost is too high it may crowd out otherwise competitive bandwidth sellers, effectively decreasing systemic efficiency by increasing effective interest rates I_c and or raising the underlying cost of bandwidth B_c. The withdrawal delay is a customizable parameter which the market ultimately will decide.

OXT is a specialized asset where major holders are not incentivized to rent out huge stake positions to unknown, unvetted entities. In that sense the rental dynamics for OXT are more likely to be similar to the rental dynamics for bitcoin ASICs, where the hashpower available for rent is a small fraction of the total. We expect the whitelist mechanism (section 4.3) to help secure any stake rental market by further discouraging stakeholders from renting to entities that are not also on the same whitelist at the risk of their own delisting. In essence this forces the penalty of delisting (incurred through the withdrawal delay) to transfer from the operator renter to the stakeholder rentier.

Burn-Weighting

We also considered burn-weighting models where stake deposits are replaced with provably destroyed OXT currency. Burn-weighting is actually equivalent to our stake-weighting model with a withdrawal delay of infinity, in which case the stake deposit is effectively burnt. The percentage position loss term x_d from equation 7 just becomes -1 (as the full position is always lost), so that equation simplifies to the attack cost condition amounting to just the sum of (burnt) stake deposits.

The same arguments concerning non-monotonicity of economic security with increasing withdrawal delay thus apply to burn-weighting (withdrawal delay of infinity). As the delay increases stakeholders lose optionality on their capital deposit, and will thus tend to demand higher effective interest rates to compensate for that lost optionality.

As burn-weighting is already a parameter mode of our current stake-weighting design, we could in the future move towards a burn-weighting model by slowly ratcheting up the withdrawal delay. There is of course risk of forks or market segmentation for clients that refuse the increase, but in theory such a change is quite possible and made easier by the decision to parameterize withdrawal delay.

Interest-Weighting

A final alternative we considered is replacing direct stake weighting with the effective interest or opportunity cost on the stake deposit over the withdrawal delay as the weighting term. The motivation behind this design is to incentivize a wider diversity of withdrawal delays by more directly compensating the staker's time-dependent lockup cost. The weighting term in interest-weighting would be something like $(1 - e^{(-w_t I_r)}) S$ where w_t is the variable operator determined withdrawal delay, I_r is a global 'interest rate' parameter, and S is the size of the stake deposit.

In this interest-weighting design, the key design parameter I_r should probably be set close to actual market interest rates or opportunity costs for OXT stake deposits. If the interest rate term I_r is much smaller than market rates, then participants are incentivized to choose withdrawal delays w_t of infinity (or their maximum), and the system decays into a form of proof of burn. If I_r is much larger than market rates, then participants are incentivized to choose very short withdrawal delays, and the system is similar to stake-weighting with short withdrawal delay.

Since a successful systemic attack will substantially lower the value of OXT and thus the value of the stake position, serious attackers effectively have extremely high interest rates or opportunity costs for OXT, as they believe it will collapse in value. Thus assuming an interest rate term I_r near the market rate, serious attackers will naturally choose very long withdrawal delays and get an effective discount in their attack cost with interest-weighting vs stake-weighting. This is because in these conditions most market participants will choose reasonable withdrawal

delays that result in a weighting term considerably smaller than 1, lowering the total stake deposit size vs stake-weighting, whereas attackers will choose infinite delay for a weighting term of 1.

Given these security concerns, the additional complexity of some unknown dynamic mechanism to adjust the global interest rate parameter I_r towards the market equilibrium and finally ethereum implementation concerns of complex weighting functions involving exponentiation and multiplication, we decided against interest-weighting.

Summary

We moved to a stake-weighting design because of the following key advantages over our earlier proof-of-work Medallion design:

1. Proof-of-work creates an additional compute burden on end users
2. Proof-of-work has far lower attack costs than stake-weighting with delay, even assuming rental markets
3. General compute rental markets already exist with far more relative liquidity than we expect will exist in any future OXT stake deposit rental markets
4. Stake-weighting captures the future discounted profits of bandwidth sellers, creating a larger baseline token market cap. This topic is explored in the next section.

4.5 Tokenomics

Stake-weighting has the distinct advantage of greater value capture than competing mechanisms for utility token systems. In this section we will briefly expound and analyze some of the relevant economic assumptions into a simple model focusing on user nanopayment deposits and node stake deposits We assume that any additional value component outside those categories (such as short term high velocity turnover of the ERC20 token itself) are relatively small in their contribution.

Market Sizing

We will start with a scenario where Pzdid has 2 million customers paying on average $5 a month, or $120 million a year in gross system revenue. For reference, the global VPN market size is expected to reach $27 billion in 2020[13].

User Deposits

We expect that most users will pre-fund their nanopayment account with a supply of OXT sufficient to pay for at least three months of bandwidth, or $15 worth of OXT in this example. VPN users are already accustomed to prepaying for months or years of service, as this has become the standard payment model in the VPN market.

The total value of user deposits in this example is thus $30 million.

Node Stake Deposits

Pzdid is a competitive bandwidth market, and as such we expect the system will eventually evolve into an approximate equilibrium where gross revenue approaches the underlying costs, which includes both the raw cost of bandwidth to suppliers and the interest or opportunity cost on stake deposit capital. Recall equations 8 and 9 from *section 4.4*:

$$R = B + I_r S \qquad (8)$$

$$S = (R - B) / I_r \qquad (9)$$

Here R is the total revenue flow, B is the seller's raw cost of bandwidth, I_r is the effective interest rate (opportunity cost), and S is the total stake deposits.

There are now a number of proof-of-stake cryptocurrency systems where holders earn interest on their stake by running nodes. The staking interest rates for each coin vary considerably based on system details, perceived exchange risks, etc. We assume an effective APR (Annual Percentage Rate) of 20% for OXT stake, which is within the typical range of rates for staking returns[14].

IP transit prices vary by location, but a reasonable median estimate is $1 per month per 1 Mbps[15], which works out to $1/month for more than 300GB of data, or about $0.003 per GB. We will use a wholesale bandwidth price of $0.01/GB. The average monthly data usage for US broadband households is 268 GB of data per month[16], so we will use 100GB/month as an estimate of per customer VPN data per month. This implies $1/month or $12/year as the per user raw cost of bandwidth, and $24 million for the total bandwidth cost term B for a year.

The total value of node stake deposits in this example is thus approximately $480 million via eq. 9.

Cryptocurrencies such as Bitcoin are mostly used as a *store of value*. The OXT cryptocurrency that powers Pzdid is a *utility token* implemented on top of the Ethereum blockchain. While it is possible that a utility token could be used as a store of value, and independently possible that the Pzdid nanopayment system could find usage outside of Pzdid , we expect the staking mechanism to capture most of the value. There are now a number of cryptocurrency systems with staking rewards and a wide variation in their APR yields. The staking ratios (value of total stake deposits over market cap) of these staking coins also varies considerably: Decred has a staking ratio of 50%[17], whereas NXT has a staking ratio of 15%[18].

5. Nanopayments

5.1 Introduction

Most layer 1 on-chain payment options today suffer from a lack of usability primarily associated with long confirmation times, low throughput, and high transaction fees. As an example, Ethereum and Bitcoin have confirmation times of 15 seconds and 10 minutes respectively, with transaction fees of roughly $0.10. [29] [30] In the Pzdid Network, we associate packet transmission (and by extension, bandwidth) with value. Thus, if the transaction fees and confirmation times for transmitting packets are as high as current layer 1 solutions offer, Pzdid ’s network economics completely break down. Simply put, the transaction fees and confirmation times associated with sending a packet should not be orders of magnitude higher than the value and propagation time of the packet itself.

Our payment scalability requirements naturally suggest the use of layer 2 micropayment solutions as the payment backbone for the network. However, as data transmission is tied closely to payment information, Pzdid ’s guarantees for bandwidth and packets must apply to payments as well. In particular, Pzdid ’s goal of reducing Internet surveillance and censorship means that both the data transmission protocol and payments protocol should additionally be censorship resistant, anonymous, and decentralized or trustless. Below, we break down these use case requirements into technical evaluation points to gauge how well both existing work and our proposed protocol solve Pzdid ’s core payment challenges.

Scalable: The system must support millions of users making frequent tiny transactions (on the order of once per second), implying negligible transaction fees per payment in expectation.

Trustless: The system should not require that participants trust particular entities such that functionality is dependent upon their specific performance and goodwill.

Anonymous: Payments should leak minimal additional information about real world identities. In addition, there needs to be deniability for all parties in the system for suspected sending, receiving, or propagation of funds [31].

Uncensorable: It should be prohibitively expensive for an adversary to censor transactions, which implies, at a high level, that it is economically or cryptographically infeasible to corrupt information or prevent its access or publication [31]. In other words, unless the majority of the network is controlled by malicious actors attempting to censor payments or packets, it should be possible to find some way of sending and receiving money without corruption to arbitrary endpoints.

In the following sections, we discuss existing payments solutions, how they fit into our evaluation framework from above, and show that Pzdid ’s payment framework can provide better guarantees for our specific use case than existing solutions can.

5.2 Existing Work and Comparisons

As suggested above, the prerequisite to transferring value associated with arbitrary amounts of bandwidth, down to potentially the packet level, is having a robust micropayment infrastructure, of which layer 2 solutions are most popular. Layer 2 solutions tie in the security of on-chain payments with protocols that don’t directly involve the main blockchain in every transaction. Theoretically, this can provide for some great benefits including lower

transaction fees, faster confirmation time, and more. Unfortunately, there are currently no production-ready micropayments solutions that are available in the ecosystem today. We explore the failures of existing schemes within the key evaluation points discussed in *section 5.1* and proceed to propose a new nanopayment protocol for stochastic value exchange.

5.2.1 Centralized Payments

Traditional financial payments are transactions settled through interparty negotiations such as that between banks or payment service providers. These settlements often take place through centralizing protocols such as ISO/IEC 7816 [32] in the case of payment cards, ACH for payroll and credit transfers[19], or NYCE [36] and SWIFT [34] for ATM transactions. Participants in these networks synchronize their local ledgers with the central network using a blend of electronic payment receipts and manual reconciliation [37].

Centralized payment systems, unfortunately, do not offer support for most of the requirements enumerated in *section 5.1*. The prevalence of fraud in the centralized financial ecosystem [38], as well as the solution to fraud, namely reversal transactions [39], each violate the principle of *trustless* operation. While responsiveness is extremely high in centralized systems, the lack of byzantine fault tolerance and interoperability between sub-systems implies that the global system is only partially available while also suffering from consistency issues. Lastly, the trusted parties that participate in and manage the payment infrastructure typically have detailed metadata about each transaction -- sender, recipient, amount and time -- and thus have all the ingredients necessary to engage in and comply with censorship and de-anonymization [40].

As noted in Pzdid 0.9.2 [6], transaction fees in centralized payments exhibit large variation, ranging from just a few cents for payment card transactions [41] to as much as $75 for international wire transfers [42]. In lieu of or in addition, many systems charge a percent fee, ranging from 3.5% for payment cards [43] to 13% for bank transfers [44]. Whereas fixed fees are generally inappropriately sized for micropayments, percentage-fee-based systems could provide a reasonable backbone for micropayments. In particular, Asia's adoption of WeChat Pay and Alipay show commercial viability of incredibly low percentage-based fees, typically between 0.0%-0.1% [45]. Unfortunately, these systems still suffer from all the centralization drawbacks mentioned prior.

F = Fully featured, P = Partially featured, N = Not featured

Scalable	Trustless	Uncensorable	Anonymous
[N, P, F]	N	N	N

5.2.2 Payment Channels

Payment channels are a newer layer two solution for scaling up the security and guarantees of traditional layer 1 blockchain systems. The Lightning Network on Bitcoin [46] was one of the first protocols to explore this type of solution. At an abstract level, most payment channels involve three steps: locking of funds in an escrow, transacting using those funds off-chain, and upon closing of the payment channel, broadcasting the final state to the escrow and paying the two channel participants.

There are a number of issues with existing payment channel infrastructure, however, that makes their use untenable for the Pzdid network. Firstly, the complexity of routing funds over payment channels is on average $O(log(n))$ hops on sending and receiving funds, where n represents the number of nodes in the network. While each hop in an end-to-end payment route is quite low in cost to the network and is concentrated primarily in routing/computational cost, the entire route incurs pairwise setup and teardown costs for the payment channel as well. An adjacent issue with this is that if one hop in the network fails to pay, it can trigger timeouts that stall the entire route. This implies an $O(c * n)$ setup and teardown complexity amortized over the average lifetime of a payment channel, where c represents the number of payment channels each node maintains. Additionally, there is a lock-up cost of funds; when funds are locked up in a payment channel, they cannot be used elsewhere. This becomes problematic when one wishes to peer with many nodes; instead of a node being able to use all of their tokens for micropayments to any of their peers, each locked token can only interact with a single peer.

Note that payment channels are typically cryptographically enforced w.r.t. the root chain using Hash Time Lock Contracts (HTLCs). The transaction fees of payment channels are also typically low. The censorability of payment channels is a bit more nuanced. In the case of the Bitcoin network, the Heilman eclipse attack analysis [47] illustrates that it is feasible, with $> 50\%$ probability, to eclipse a Bitcoin node using a botnet of only 400 IP addresses. To apply this attack to payment channels, a node must be unable to communicate with the larger L1 network. This depends heavily on the way peering is actually handled, and thus the complexity of eclipse attacks differs depending on the L1 platform. As for anonymity and privacy, unfortunately those two properties are very limited in current payment channel technology.

F = Fully featured, P = Partially featured, N = Not featured

Scalable	Trustless	Uncensorable	Anonymous
F	P	P	N

5.2.3 Probabilistic Micropayments

The concept of probabilistic micropayments was introduced by Wheeler [48] and Rivest [49] in the late 1990s as a way of reducing the impact of transaction fees on traditional micropayments. Pass and Shelat [50] extend this idea in MICROPAY1 to blockchain based payment systems to provide the same benefits on top of a decentralized system. The core idea of this class of micropayments is similar to that of payment channels: amortizing the cost of transaction fees across numerous transactions. The core mechanism in blockchain-backed probabilistic micropayments, however, is not an HTLC, but rather the use of a lottery-based payments. In such a system, a payment of $\$X$ is actually sent as a "lottery ticket" with value $C * \$X$ and a probability of winning of $\frac{1}{C}$ so that the expected value of the ticket is $C * \$X * \frac{1}{C} = \$X$.

The scheme can generally be described as follows:

A wants to pay B
A deposits some currency to a Bitcoin escrow address h_E of a newly generated key
B generates a random number R_B and transmits a hidden signed commit to A
B also sends a recipient address h_B to A

A generates a random number R_A and signs it in plaintext, along with payment information, and transmits it to B

If $R_A \oplus R_B$ ends in 00 and R_B matches the hidden signed commit, then the ticket is a winner and the escrow pays out to B

This scheme, by design, theoretically is scalable with negligible transaction fees (as it's almost entirely off chain). Unfortunately, in practice, most existing schemes rely on a centralized intermediary somewhere in the protocol, so they are not trustless. Additionally, w.r.t. censorship resistance, the same problems with eclipsing in the payments channel sub-section appear here. The biggest difference between probabilistic micropayments and payment channels is O(1) payment routing complexity for probabilistic micropayments.

F = Fully featured, P = Partially featured, N = Not featured

Scalable	Trustless	Uncensorable	Anonymous
F	P*	P*	N

** Due to limitations of existing implementations*

5.3 Pzdid Nanopayment Scheme

The Pzdid nanopayment scheme is strongly motivated by the concepts in the MICROPAY1 scheme by Pass and Shelat [50] briefly described in *section 5.2.3.* The philosophy of our payments system attempts to make reasonable iterations from the MICROPAY1 scheme, particularly with economic scalability of the system at a negligible security cost. Towards that end, we created a protocol aiming to satisfy the requirements of a scalable, trustless, uncensorable, anonymous payment system.

With these properties in mind, we describe the Pzdid nanopayment scheme. To do so, we offer the following definitions:

Actors:

Sender: The sender of a nanopayment. The Sender is expected to have an Ethereum account and the ability to connect to some Ethereum node to setup and funds to their nanopayment account. The Sender submits payments by sending Tickets (defined below) to the Receiver, after receiving a message containing the receiver's hash commitment and destination account.

Receiver: The receiver of a nanopayment. The Receiver needs an Ethereum account and access to an Ethereum node. The Receiver generates a hash commitment and sends that along with their destination account id to the Sender, and then receives one or more Tickets from the Sender. The Receiver is responsible for ensuring the payment parameters received by the Sender are correct, and that they have the required funds available.

Payment/Membership Smart Contract: The smart contract responsible for Settling the payment process for any Winning Ticket also enforces the crypto-economic incentives against frontrunning, griefing, double spends, and other bad behavior on the part of the Sender.

Messages:

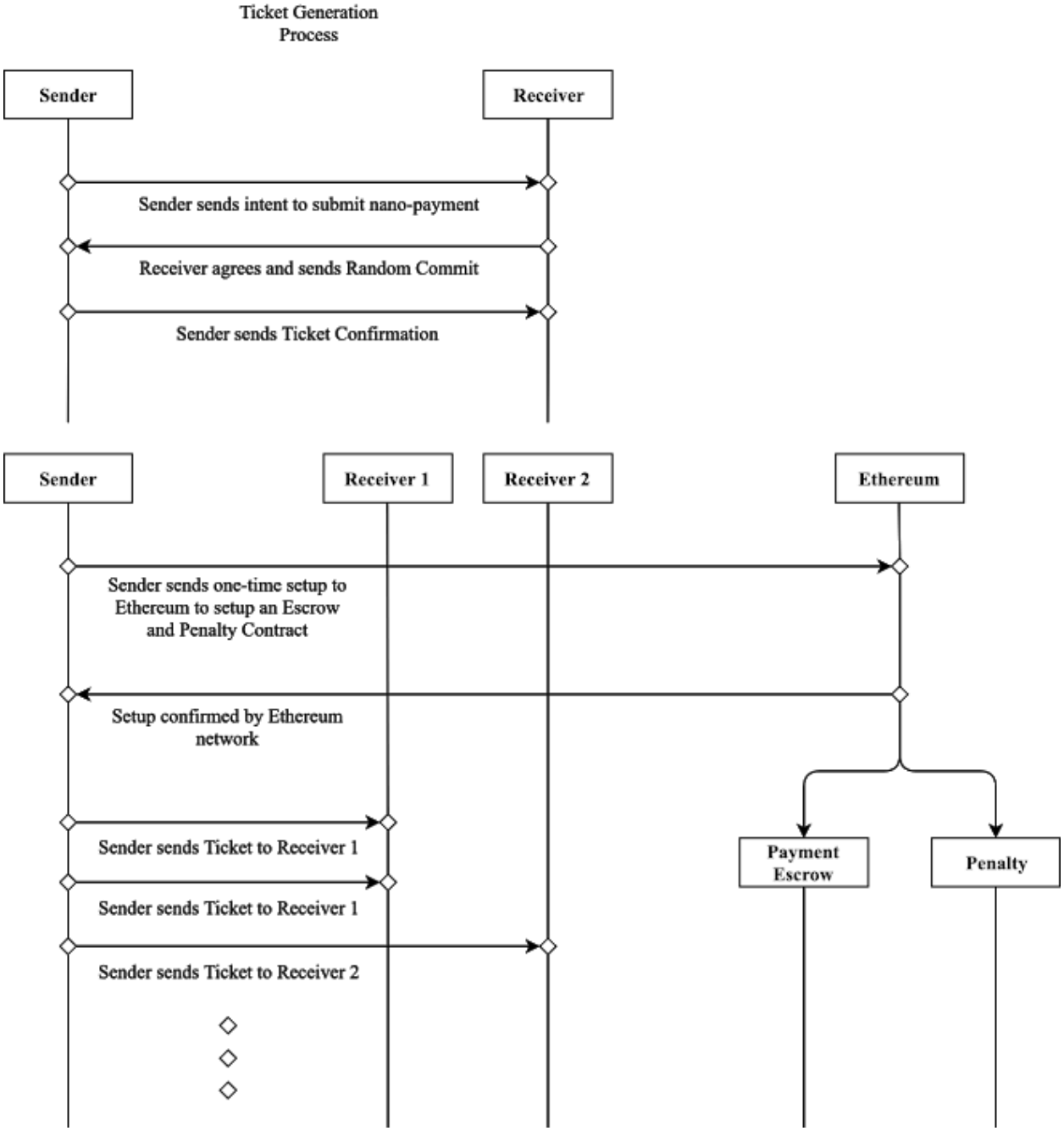
Random Commit: The commit message that the Receiver of a Ticket first sends to the Sender in order to commit to a randomly generated number. This commit hides the random number itself through a hash function.

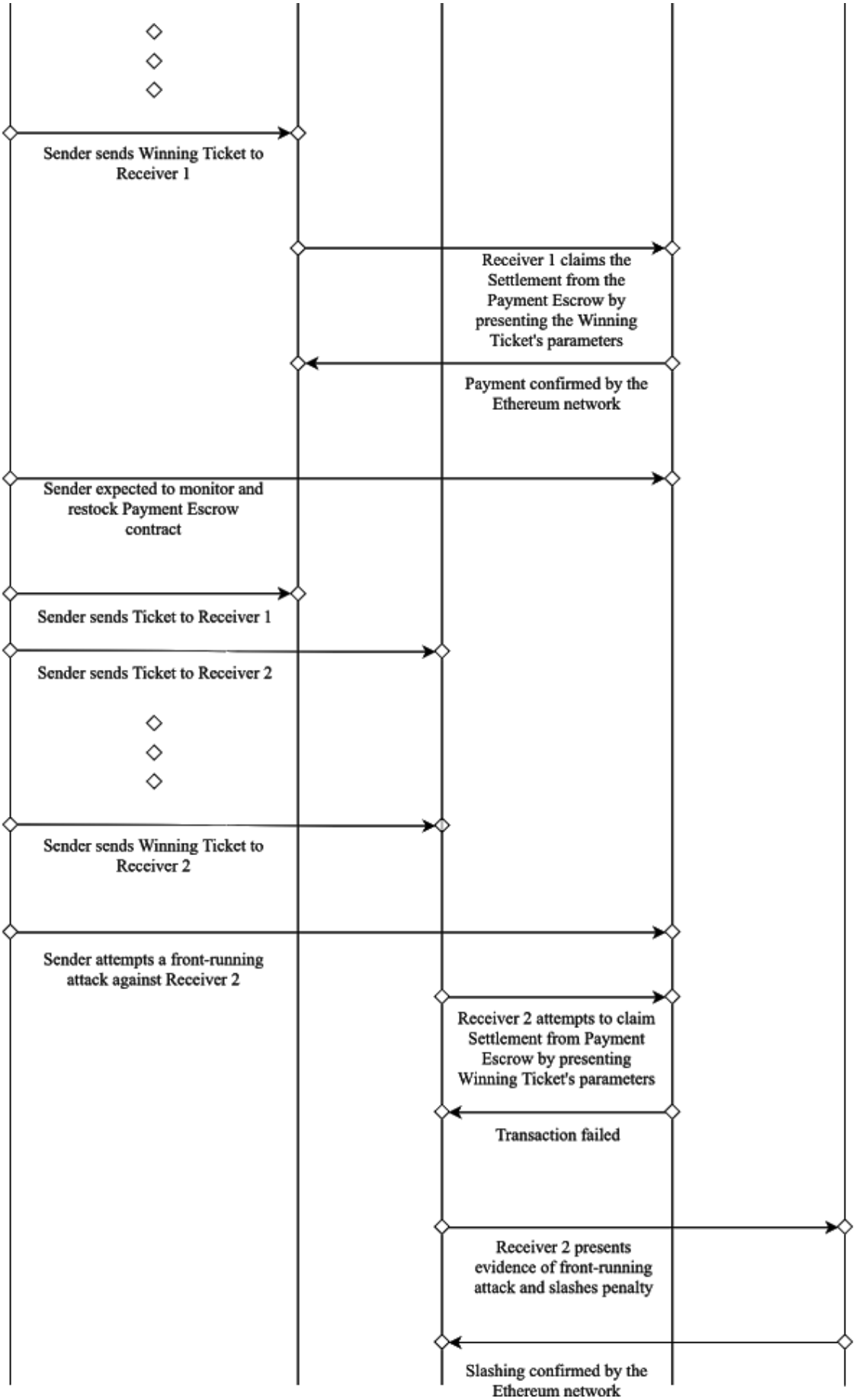
Ticket: The message that the Sender sends back to the Receiver to complete the interactive Ticket Generation process. This includes the sender's random number and a signature which confirms the key fields of the completed nanopayment. Note that a Ticket's effective value is its expected value. Its true redemption value is either the face

value agreed upon by the Sender and Receiver if the Ticket is a Winning Ticket, or 0. A Ticket is a Winning Ticket if and only if the Ticket Generation process creates a random number that satisfies the settlement conditions.
Winning Ticket: A completed nanopayment that satisfies the conditions to settle at the given face value, in particular, containing a random number that satisfies the probability of winning. This is the message that is broadcast to the Ethereum network to claim settlement from the sender's Payment escrow or used to prove Griefing.

Processes:
Ticket Generation/Sending: The process of "sending" a ticket, or more accurately, generating it, through an interactive process. The Receiver first sends a Random Commit to the Sender to begin the random number generation process. The Sender then sends back a Ticket that includes the remaining information for the Receiver to generate the Ticket, including the sender's random number.
Settlement/Redemption: The process of redeeming, settling, or cashing in a Winning Ticket. The Winning Ticket is first generated by having the Receiver sign the received information, and then is broadcast to the Ethereum network. The Payment Contract will then disburse the funds from the Payment balance into the Receiver's address.

Below is a program flow that illustrates how payments are delivered between a Payer and Receiver using the Pzdid nanopayments scheme.


Ticket Generation Process
Sender
Receiver
Sender sends intent to submit nano-payment
Receiver agrees and sends Random Commit
Sender sends Ticket Confirmation
Sender
Receiver 1
Receiver 2
Ethereum
Sender sends one-time setup to Ethereum to setup an Escrow and Penalty Contract
Setup confirmed by Ethereum network
Sender sends Ticket to Receiver 1
Sender sends Ticket to Receiver 1
Sender sends Ticket to Receiver 2
Payment Escrow
Penalty


Sender sends Winning Ticket to Receiver 1
Receiver 1 claims the Settlement from the Payment Escrow by presenting the Winning Ticket's parameters
Payment confirmed by the Ethereum network
Sender expected to monitor and restock Payment Escrow contract
Sender sends Ticket to Receiver 1
Sender sends Ticket to Receiver 2
Sender sends Winning Ticket to Receiver 2
Sender attempts a front-running attack against Receiver 2
Receiver 2 attempts to claim Settlement from Payment Escrow by presenting Winning Ticket's parameters
Transaction failed
Receiver 2 presents evidence of front-running attack and slashes penalty
Slashing confirmed by the Ethereum network

There are three key things to note with this program flow. Firstly, there is only a one-time setup for a Payer, keeping setup costs extremely low relative to other existing solutions. While this raises potential issues for double spends and frontrunning, we show both mathematically and empirically that this is highly unlikely later in the paper. Secondly, each Receiver interacts with the same Payment escrow and Membership contract, keeping setup costs for each individual Sender-Receiver pair trivially low. Additionally, this means funds used to pay different Receivers do not need to be locked or split between the Receivers, which allows for less money to be locked up in illiquid escrows meant for collateralizing the payment channels. This happens due to the statistical multiplexing phenomenon, which we see very often in networking. Finally, all nanopayments occur off-chain, keeping the efficiency guarantee, but they delegate trust back on-chain to deal with Settlements, ultimately removing the reliance on third parties that previous probabilistic micropayment methods suffered from.

We compare the features of the Pzdid nanopayment scheme with existing micropayment schemes below. We further justify these claims in the following sections and the Appendix.

F = Fully featured, P = Partially featured, N = Not featured

Payment Solution	Scalable	Trustless	Uncensorable	Anonymous
Centralized	[N, P, F]	N	N	N
Payment Channels	F	P	P	N**
Probabilistic Micropayments	F	P*	P*	N**
Pzdid Nanopayments	F	F	F	N**

** Due to limitations of existing implementations*

*** Can be addressed with mixing, one-time addresses, etc. More discussed in section 5.8 on Anonymity*

n = nodes in L2 network

C = average # of connections per node

Payment Solution	Routing Complexity	Network Setup Complexity	Fund Distribution Factor*
Centralized	N/A	N/A	N/A
Payment Channels	$log_C(n)$	C	$\frac{1}{C}$
Probabilistic Micropayments	1	C	$\frac{1}{C}$
Pzdid Nanopayments	1	1	1

** Denotes what the fraction of total funds each peer can transact with. A lower fraction generally results in a lower network throughput as a whole.*

5.3.1. Differences from MICROPAY

While the general scheme for Pzdid 's nanopayment protocol is similar to MICROPAY[40], in the Pzdid scheme, we make a few changes to the underlying assumptions in order to introduce certain efficiency benefits. In addition, these assumptions will allow us to introduce an implementation that maintains the theoretical scalability and censorship resistance behind the philosophy of the original scheme.

In the Pzdid nanopayment scheme, we change the following assumptions:

1. CHANGE: Each Payment escrow can only be used by one Receiver to avoid double spends
    a. TO: Each Payment escrow can be used by multiple Receivers to redeem Winning Tickets
2. ADD: There must be a way to mitigate depletion of funds by two distinct Receivers
3. CHANGE: Use Bitcoin scripting
    a. TO: Use Ethereum smart contracts and their supporting underlying cryptographic functions
4. CHANGE: Use a mutually trusted third-party to deal with Payment escrows
    a. TO: Use an Ethereum-based smart contract to deal with Payment escrows

We discuss how these changes affect security, double-spends, frontrunning, and more in section 5.10.

5.4 Pzdid Token (OXT)

The Pzdid Token (OXT), is a new ERC20 compliant token with a fixed supply of one billion units, and standard sub-divisibility down to 18 decimal places, like ETH. There is no inflation. The possibility of contractual penalty mechanisms that 'burn' currency such as used in nanopayment accounts to prevent double spending (section 5.10) create the potential for some small additional deflationary pressure.

Using a new custom token as the currency for the Pzdid Market provides economic incentive benefits that would not be possible if we used generic currencies such as ETH. More specifically, requiring that large providers stake large amounts of a custom utility currency specific to our market creates stronger incentive alignment effects than using a generic currency as the provider's behavior will more strongly affect the price of the custom market token and thus the value of their stake positions. If we instead used a generic currency such as ETH, this correlation would be very weak, as the health of the Pzdid market would have much less expected impact on the price of ETH.

5.5. Pzdid Gas Costs

Our current open source solidity implementation of the key ticket redemption function uses about 100K gas when called on a winning ticket, which includes the cost of the underlying ERC20 transfer (the function is only called on winning tickets).

5.6 Censorship Resistance

The Pzdid payment protocol inherits Ethereum's censorship resistance, which is similar to that of other blockchain cryptocurrency protocols. The nanopayment protocol involves only direct communication between the sender and the receiver during normal operation on non-winning tickets. Only winning tickets require the receiver to submit a transaction onto the Ethereum blockchain, so Pzdid nanopayments have the same censorship resistance as regular Ethereum transactions.

Censoring all of Pzdid 's specific Ethereum transactions (or all of the Pzdid redemption transactions for a particular receiver) would require a majority of miners to agree to ignore all winning blocks containing these Pzdid transactions. We consider this scenario highly unlikely due to the high profit risk or cost and the decentralized

nature of the Ethereum mining community. A limited form of partial censorship could be achieved if some fraction of Ethereum nodes refused to include Pzdid transactions in their winning blocks, but that would only increase transaction fees in proportion to 1 / (1-X), where X is the relative hashpower of the censoring group.

Note that, like in *section 5.2.3* on payment channels, eclipse attacks are potentially harmful, especially if a Payer or Receiver is running a full node and relying on trust to that full node in order to submit transactions to the network. However, in the case of Pzdid 's nanopayments, Payers and Receivers do not need to be running full nodes in order to participate in the Pzdid nanopayment network. Additionally, any party that is running a node can also submit transactions to peers that they trust, or well-known public peers, to ensure that their transactions are not censored. This is one of the key benefits of Pzdid 's scheme and associated implementation over existing L2 payment channel schemes.

5.8 Anonymity

Pzdid nanopayments are only pseudo-anonymous: during redemption of winning tickets the receiver posts the normally private offline client-server payment information on chain, creating a permanent public record. Losing tickets are not posted, so they reveal payment information only to the recipient. This reduces payment information leakage, but after weeks to months of use, winning tickets will still accumulate and leave a public information trail that links a user's public account key(s) with some of the Pzdid providers they have paid. The payment ticket does not reveal the particular server the client was connected to, only a public key of the provider, but more sophisticated attackers could pose as users to build up a model of any server's public keys and physical addresses.

For most users this small amount of information leakage is not a serious problem, but users desiring stronger payment privacy can take appropriate steps to break the linkages between their Ethereum accounts and real world identity before funding their nanopayment account(s) (using mixing services, conversion to anonymous cryptocurrencies, etc). For multi-hop routes, the Pzdid client can use separate nanopayment accounts and public keys for each node in the circuit to protect against route inference from on-chain payment history (assuming appropriate prior disentangling of the multiple funding accounts).

5.9 Scalability Analysis

The Pzdid nanopayment system is a layer 2 scaling solution that can provide many orders of magnitude higher transaction throughput than existing layer 1 blockchain payment systems, but ultimately the maximum viable transaction throughput is a multiplier on that of the underlying layer 1 foundation. There are three main sources of on-chain transactions in our nanopayment system:

1. Users deposits/withdrawals into/from nanopayment accounts
2. Seller stake deposits/withdrawals into/from stake registry accounts
3. Seller redemptions of winning tickets

We will assess scalability first from the perspective of transaction fees, and then from the perspective of the fundamental transaction throughput limits of Ethereum.

Typical average Ethereum transaction fees are on the order of $0.05 [51] for a standard transaction with a gas cost of ~20k gas. Typical VPN users prepay for 6 months to a year or more, so we assume that most Pzdid users will typically 'prepay' by depositing on the order of $10 to $50 in their nanopayment account to fund multiple months of bandwidth purchase. Thus the transaction fees for user deposits and withdrawals are only a small overhead burden, even assuming a larger gas cost. Transaction fees for bandwidth seller stake deposits/withdrawals are even less

significant: if typical sellers have at least thousands of clients, monthly revenue exceeding $1000, and only add or remove stake once per month leads to transaction fee overhead of less than 0.1%.

The overhead for ticket redemption transaction costs varies. The expected value of a nanopayment is the win probability multiplied by the face value, which allows flexibility in trading off between variance and transaction fees. Using a low win probability and high face value reduces transaction fee costs by lowering the expected number of winning tickets per unit time at the expense of increased variance. Conversely, high win probability of low face value tickets reduces variance at the expense of more frequent winners, redemptions and transaction fees.

The current Pzdid smart contract payment redemption function uses about ~100k gas, which translates to a transaction fee of ~$0.02-$0.20 reflecting current prices. Using the assumption that a 5% transaction fee overhead is reasonable leads to $4 face value tickets. A user who deposits $40 in their nanopayment account for 4 months of bandwidth usage will then on average issue 10 winning tickets over the 4 month period.

We can model the depletion of that balance using a binomial distribution. Assume tickets are issued at an amortized rate of about 1 per second (this usage pattern is not an essential feature of the analysis but is used for the purpose of illustration) or about 10 million tickets per 4 month period, with a win rate of 10^{-6}. With a pool of 10 winners there is a ~1.8% chance that the account will deplete in only 2 months or less, i.e. more than twice as fast as expected. Conversely, there is only a ~0.6% chance that the account will last 8 months or more.

To minimize transaction fees in this example, we could reduce the win rate by 10x and use $40 face value tickets leading to an expectation of only 1 winning ticket per 4 months. This would reduce the transaction fee overhead down to 0.4%. However, with these settings the risk of depletion is enormous: there is now a ~30% chance that the account would be depleted after 2 months or less.

Transaction throughput in the Ethereum blockchain depends on the transaction gas cost (a fixed property of the transaction's compiled EVM code), as well as the block gas limit and the block production rate, which both vary over time. Our ticket claim function uses around 100k gas. Ethereum currently supports 10 million gas per block [52] produced at a rate of one block per 13 seconds [53], leading to a throughput of about 7 tps (or 18 million transactions per month) for 100k gas transactions. Using the earlier example of roughly 2.5 winning tickets per user per month leads to a maximum scaling limit of about 7 million users, assuming Ethereum was used solely for Pzdid transactions.

Scaling Pzdid 's nanopayment system to tens of millions of users and beyond will require the deployment and utilization of scaling improvements in the underlying layer 1 blockchain, such as Ethereum 2.0 with sharding, or migration to a new layer 1 solution with higher base throughput.

5.10 Cryptoeconomic Methods for Preventing Griefing

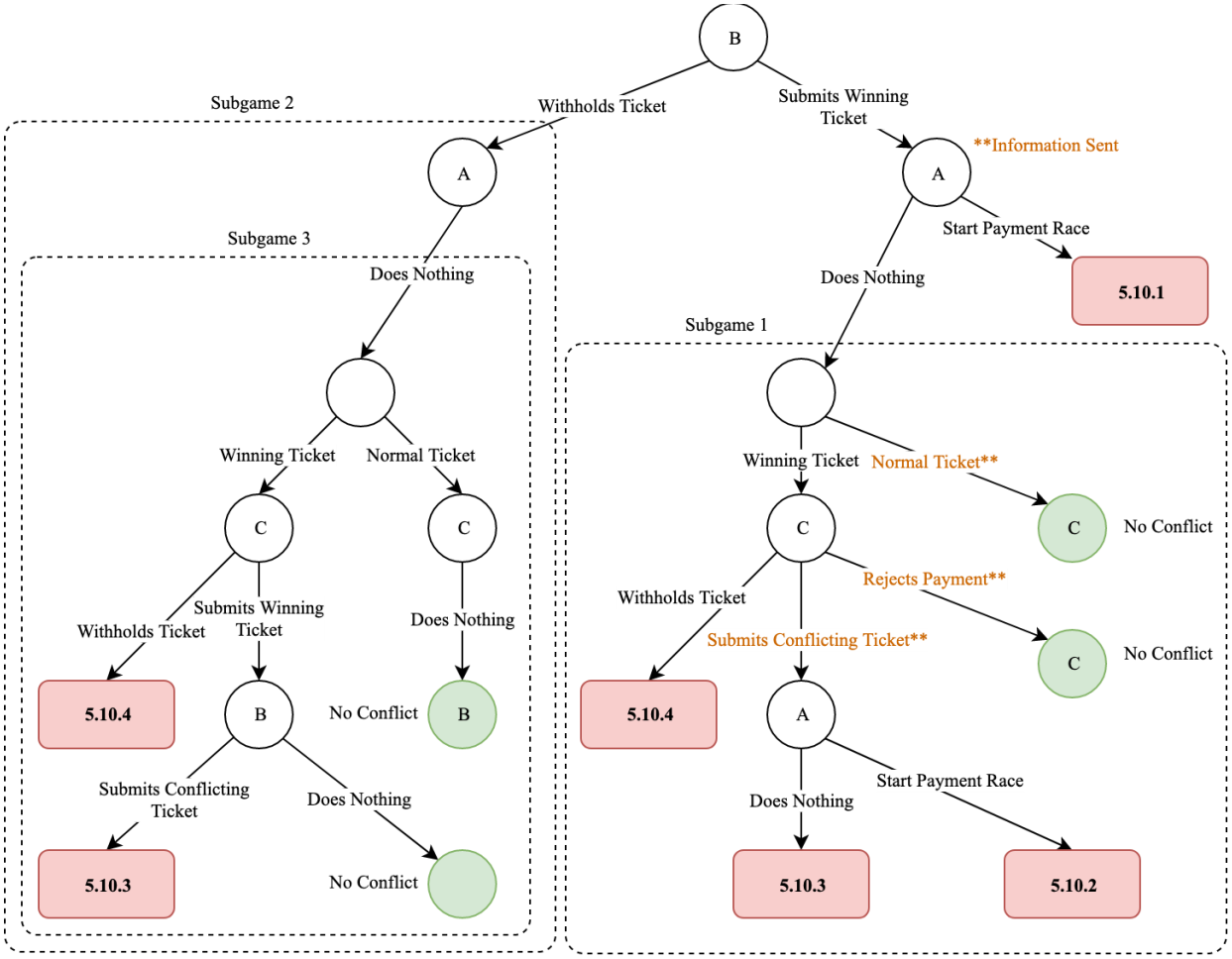
As mentioned in *section 5.3.1*, one of the key differences between existing probabilistic micropayment schemes and Pzdid nanopayments is the introduction of a need for cryptoeconomic incentives to prevent malicious attacks. This is because every Receiver that a particular Payer *A* is sending nanopayments to is redeeming winning tickets from the same Payment smart contract (which belongs to *A*). We describe efficiency ramifications in *section 5.3.*

The biggest problem this design introduces centers around one idea: a Receiver receiving a winning ticket from the Payer but not being able to settle due to a lack of funds in the Payment escrow. This happens explicitly when there is not enough balance in the Payment escrow to cover all the Winning Tickets that will be submitted. We outline the

extensive form game tree below to outline how different attack cases may arise. We specify the following assumptions:

1. Our goal in using the extensive form game tree is not to find optimal strategies, but rather avoid bad ones
2. A "bad" strategy from a network governance perspective is any strategy that causes a legitimate Winning Ticket to not be paid out in full
3. Thus, we do not outline all strategies that lead to legitimate payouts, and instead focus on actions that have the potential to lead to a bad strategy
4. We provide no assumptions on whether actors are malicious, benign, or adhere to or deviate from protocol
5. We take the minimal assumption that rational actors will not choose attacks where the cost of an attack is greater than the expected net benefit

With these assumptions (and lack thereof) in place, our goal is to find potential bad strategies and introduce incentive models to avoid those bad strategies. We assume that actors only act when there is enough in the Payment escrow to pay out a Winning Ticket, as if this is not true, the incentives and disincentives for both Payers and Receivers break down - there's no reason for anybody to use the payment system if it is not functional. We outline the simplified version of the extended form game tree below. While the tree is simplified, we will show that other branches of the tree collapse down to the four failure cases that are listed in the diagram in *section 5.10.5*.



A = Payer *B, C = Receivers*

As we can see, there are a number of steps that participants in Pzdid 's nanopayment scheme could follow that expose bad strategies. We analyze each one below to show how we can prevent such bad strategies from being exposed through local incentives for each subcase.

Note the "information sent" after B submits a Winning Ticket refers to the propagation of knowledge of B's Winning Ticket to the network. There are a number of decisions that depend on the existence of this knowledge, even though the receipt of this knowledge is not a decision unto itself. In particular, the random node at the top of *subgame 1*, is generating payments that are sent to C -- an honest node C that knows of B's Winning Ticket will immediately reject all payments from A. This is the implementation that Pzdid has provided -- if there is knowledge of an existing claim to A's payments, then all further packets should be rejected. While this is the implementation we provide, we note that, even if a benign (or malicious actor) does not follow this, or has not yet received knowledge of B's Winning Ticket, the below vulnerability analysis provides cryptoeconomic incentives against bad strategies.

5.10.1 Payer Single-Entity Frontrunning Attack

This attack, commonly known as frontrunning, occurs when a Payer attempts to avoid Settlement by attempting to submit a Winning Ticket to the Payment escrow before B can settle, thereby avoiding payment to B. The key to disincentivizing this attack is to make sure that the penalty received for attempting frontrunning outweighs the benefit received from frontrunning. We list definitions below.

B_{Escrow} = *Amount in Payment escrow balance*
$B_{Membership}$ = *Amount in Membership balance*
V_{Ticket} = *Face Value of Ticket*
r_{win} = *Win rate of Ticket*
V_{txn} = *Cost of transaction*
$V_{Ticket} - V_{txn}$ = *Settlement value of Ticket*

In the case of a frontrunning attack, our goal is to mitigate the existence of a bad strategy by disincentivizing the Payer enough such that they will not rationally choose this attack. In particular, it must cost the Payer *more* to conduct this attack than it would cost them to simply pay out the ticket. Worded alternatively, the utility the Payer receives from conducting this attack must be *less* than it would cost them to simply pay out the ticket.

$$V_{Ticket} - V_{txn} < B_{Membership} - V_{txn}$$

So long as the inequality above holds, which is easy to specify and verify on-chain, it is possible to disincentivize a rational Payer from choosing this second case by slashing the membership deposit and thus making it more expensive to execute a bad strategy than it is to simply pay the Receiver.

5.10.2 Payer Multi-Entity Frontrunning Attack

In the case where multiple Receivers receive Winning Tickets in quick succession, some of them may begin Settlement before any of them knows that somebody else is also claiming the Payment escrow. In this case, it is possible for the Payer to conduct a frontrunning attack that even still circumvents the above inequality. If there are n Winning Tickets that are submitted in quick succession, the inequality from above to prevent multi-entity frontrunning becomes:

$$Payout = n * (V_{Ticket} - V_{txn}) < B_{Membership} - V_{txn}$$

Unfortunately, there are two problems with maintaining this inequality. Firstly, if n is unbounded, the number of tokens that must be locked in the Penalty balance begins to scale linearly with the number of Receivers and size of payments, making the Pzdid nanopayments scheme no more fund-distribution efficient than other payment methods. Secondly, it increases the potential harm caused by Winning Ticket collisions that are completely benign, as we will cover in *section 5.10.3*. Thus, there seems to be no logical way forward without either violating our system design assumptions or introducing incentives that disincentivize good behavior. Luckily, if we introduce a slightly stronger assumption, we can solve this dilemma.

We assume that a rational actor will not choose a bad strategy if the expected value of the utility gained from playing the bad strategy is dominated by other strategies. With this assumption we can place a bound on the risk that the system can take and ensure that the expected cost of front-running attacks is low in order to minimize locked funds. We can then use this bound to introduce a better bound on $B_{Membership}$. To do so, we introduce the following assumption, along additional definitions:

Δ = *average time difference between when A submits the Settlement for a Winning Ticket and B is aware of it*
r_{OXT} = *amortized rate of OXT per second Payer sends to Receiver*
V_{Δ} = *value of OXT transferred between Payer and Receiver over time* Δ
N_{Δ} = *number of Tickets sent between Payer and Receiver over time* Δ
r_{Ticket} = *E(number of Winning Tickets per second)*

We derive the following from our definitions:

$$V_{\Delta} = r_{OXT} * \Delta$$
$$N_{\Delta} = \frac{V_{\Delta}}{Expected\ Value\ of\ Ticket} = \frac{V_{\Delta}}{V_{Ticket} * r_{win}}$$
$$r_{Ticket} = \frac{r_{OXT}}{V_{Ticket}}$$

To get the probability of a Winning Ticket collision with n total payments peers given W = *one Winning Ticket has been found*, we do the following:

$$P(c\ collisions|W) = C_c^{n-1} P(\ specific\ Receiver\ collision|W)^c P(specific\ Receiver\ no\ collision|W)^{n-c-1}$$
$$P(specific\ Receiver\ no\ collision|W) = P(not\ a\ Winning\ Ticket)^{N_{\Delta}} = (1 - r_{win})^{N_{\Delta}}$$

This means that as the win rate r_{win} decreases, the probability of a collision decreases as well. Thus, the intuitive approach to selecting payment hyperparameters to prevent collisions is simply to decrease win rate. Let's see how we can bound the Membership balance with this approach as well:

$$P(specific\ Receiver\ collision|W) = 1 - P(specific\ Receiver\ no\ collision|W) \approx 1 - e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})}\ if\ r_{win} \ll 1$$
$$P(c\ collisions|W) \approx C_c^{n-1}(1 - e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^c (e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^{n-c-1}$$

Now that we have a probability of collision, we can provide a bound on expected loss from a front-running attack:

$$E(payout) \approx V_{Ticket} + \sum_{i=1}^{n-1}(P(i\ collisions|W) * i * V_{Ticket})$$
$$E(payout) \approx V_{Ticket} + \sum_{i=1}^{n-1}(C_i^{n-1}(1 - e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^i (e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^{n-i-1} * i * V_{Ticket})$$

So long as $E(payout) < B_{Membership}$, then on average, it will not be good to attempt a front-running attack. To minimize the $E(payout)$, and thus decrease the amount of funds we must keep locked up while maintaining cryptoeconomic disincentives against bad strategies, the hyperparameter strategy above applies here as well. We simply must select a large V_{Ticket}, and correspondingly, a low r_{win}. Thus, the intuitive approach to our payment hyperparameter choices from above, simply lowering the ticket win rate, effectively provides a provable bound for funds locked in the Membership balance that can be lowered to effectively be constant with respect to the number of Receivers.

Note, however, that this model does not count for the Payer being able to effectively monitor many, if not all Receivers. If this is possible, then it is possible for the Payer to avoid attempting front-runs against situations where it is not profitable to. As a defense against this, the hyperparameter strategy from above already makes those cases far less likely: as the probability of collisions becomes vanishingly small, the expected cost of a collision also becomes vanishingly small, and thus becomes negligible.

Below, we show some empirical choices for bad and good payment hyperparameters, as well as their resulting collision rate. Note that we measure collision rate with respect to the existence of even a single collision. The analysis in this section is primarily to protect Receivers from frontrunning attacks, meaning within the context of this section, it is up to the Receiver to only accept payments that conform to safe parameters

Parameter	Δ	r_{OXT}	r_{win}	V_{Ticket}	Collision Rate $n = 2$	Collision Rate $n = 10$	Collision Rate $n = 100$
Bad Strategy	300s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-2}	0.12 OXT	~ 0.747%	~ 6.527%	~ 52.41%
Okay Strategy	30s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-3}	1.2 OXT	~ 0.0075%	~ 0.0675%	~ 0.740%
Good Strategy	3s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-4}	12 OXT	~ 0.0000075%	~ 0.000675%	~ 0.00742%

5.10.3 Multi-Entity Payment Races

Multi-entity payment races are Winning Ticket collisions that are not malicious on the Payer's part. These payment races occur naturally. However, there are two cases where these payment races can be reached - one of them is outlined in *subgame 1* and the other in *subgame 2*. We outline their cases below and discuss how to prevent them.

Subgame 1: Unintended Payment Race

When an unintended payment race happens, we can use the collision analysis from *section 5.10.2* to select hyperparameters that minimize unintended payment races. While it is not possible in an asynchronous setting to completely prevent payment races, the risk associated with them is as follows:

$$P(Any\ collision\ per\ second|W) = r_{Ticket} * P(Any\ collision \mid W)$$
$$P(Any\ collision\ per\ second|W) = \frac{r_{OXT}}{V_{Ticket}}(1 - P(0\ collision \mid W)) \ .$$
$$P(Any\ collision\ per\ \ second|W) = \frac{r_{OXT}}{V_{Ticket}}(1 - C_0^{n-1}(1 - e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^0(e^{(\frac{-r_{OXT} * \Delta}{V_{XTicket}})})^{n-1}) \ .$$
$$P(Any\ collision\ per\ second|W) = \frac{r_{OXT}}{V_{Ticket}}(1 - (e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^{n-1}) \ .$$
$$E(penalty|W)\ per\ second = P(Collision\ per\ second|W) * B_{Membership}$$
$$E(penalty|W)\ per\ second = \frac{r_{OXT}}{V_{Ticket}}(1 - (e^{(\frac{-r_{OXT} * \Delta}{V_{Ticket}})})^{n-1}) * B_{Membership}$$

The risk that the Payer takes in unintended payment race losses is shown below in a few examples. We calculate $B_{Membership}$ from the bounds in *section 5.10.2*.

Parameter	Δ	r_{OXT}	r_{win}	V_{Ticket}	Payment Race Penalty per Second $n = 2$	Payment Race Penalty per Second $n = 10$	Payment Race Penalty per Second $n = 100$
Bad Strategy	300s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-2}	$0.12\ OXT$	$2.258 * 10^{-8} \frac{OXT}{s}$	$2.089 * 10^{-7} \frac{OXT}{s}$	$2.735 * 10^{-6} \frac{OXT}{s}$
Okay Strategy	30s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-3}	$1.2\ OXT$	$2.251 * 10^{-10} \frac{OXT}{s}$	$2.026 * 10^{-9} \frac{OXT}{s}$	$2.236 * 10^{-8} \frac{OXT}{s}$
Good Strategy	3s	$3 * 10^{-6} \frac{OXT}{s}$	10^{-4}	$12\ OXT$	$2.250 * 10^{-12} \frac{OXT}{s}$	$2.025 * 10^{-11} \frac{OXT}{s}$	$2.228 * 10^{-10} \frac{OXT}{s}$

Observe that the collision analysis in the previous section was primarily to protect Receivers against frontrunning, resulting in Receiver-driven incentives for choosing good hyperparameter strategies. Note that in unintended payment races, the Payer is mistakenly punished for a race they had no control over. In the worst case above, ~1% of fees can be taken due to poor hyperparameter choice. With good strategies, this fee becomes negligible. Thus, the existence of the unintended payment race creates Payer-driven incentives for choosing good hyperparameter strategies as well.

Subgame 2: Withholding Attack by Receiver

If a Receiver withholds a Winning Ticket and broadcasts it immediately after another Receiver has submitted a Winning Ticket, the Receiver can force a slash on the Payer, opening up a bad strategy that hurts the Payer. Thus, our goal is to sufficiently disincentivize withholding such that a Receiver does not do this. Recall from *section 5.10.1* that an invariant must hold to discourage single-entity frontrunning, namely $V_{Ticket} < B_{Membership}$. Thus, if a Receiver wishes to grief, the amount of damage caused is, at initial examination, higher than the cost of inflicting damage. One solution could be to decrease the burned amount as time elapsed increases. However, this begins to interfere with our cryptoeconomic invariants from above that prevent Payer-initiated attack vectors.

Thus, our next best course of action is to attempt to reduce the amount of damage that a potential withholder could inflict. Note that our analyses from above allow us to calculate the probability of a collision given a Winning Ticket has already been found! In other words, if we only consider a Winning Ticket valid for the duration Δ from above, the Payment Race penalty over time is the same as it is in *subgame 1* (see above for empirical analysis). Note that, not only is the expected damage rate for a withholding attack vanishingly small, the damage itself is only valid over duration Δ. Note that this validity period is variable, and can actually go lower than the expected time for a claim to be executed to further lower the chance of damage.

Thus, the expected loss from withholding attacks is vanishingly small as good parameters are chosen. With these vanishingly small expected losses, the ratio of expected cost to an attacker relative to expected loss to the victim is extremely high, which from our rational attacker model, prevents a bad strategy from existing due to withholding attacks. As mentioned in *subgame 1*, this further create Payer-driven incentives for choosing good hyperparameter strategies.

5.10.4 Withholding

Note that this is, rationally, not a valid strategy for any benign or honest Receiver. They receive vanishingly small expected tangible benefit for withholding, and at worst, are giving up Winning Tickets and the payouts associated with them. There are, in fact, only two cases where we care about withholding: the descent of withholding into

recursive subgames, and withholding attacks. All other cases are benign to the network and don't have the opportunity of opening up bad strategies - in fact, withholding only causes direct economic harm to withholders. Thus, as mentioned in 5.10.3 subgame 2, we simply need to include an expiration time for all Winning Tickets to keep the expected damage of withholding low and incentivize Receivers to Settle their Winning Tickets as quickly as possible.

5.10.5 Recursive Subgames

The extended form game in this section has no limits on the number of actors, nor the number of actions that can occur. We can note, however, that each of the failure cases from above either invalidates the base assumption of the extended form game (the availability of enough funds in the Payment escrow to cover a single Winning Ticket) and thus exits the framework of the game, or leads recursively to subgame 1 or 2. We enumerate this mapping below.

- 5.10.2 leads to an invalidation of the base assumption. If the rest of the network has not arrived at that conclusion yet, node 5.10.2 leads to the entry node in both subgame 1 and subgame 3 with arbitrary Receivers B and C.
- 5.10.3 also leads to an invalidation of the base assumption. If the rest of the network has not arrived at that conclusion yet, node 5.10.3 leads to the entry node in both subgame 1 and subgame 3 with arbitrary Receivers B and C.
- 5.10.4 leads to the entry node of subgame 2

Note that each of the failure cases in a subgame could lead to a recursive case of the existing game tree. Any of the benign paths (green nodes) could open up additional potential attacks by recursively descending into any of the subgames (simply with receipt of a new ticket by an arbitrary actor). However, these subgames will always ultimately lead to (now infeasible) bad strategies, or terminate on a benign path.

5.10.6 Summary

In conclusion, based on our cryptoeconomic model from above, we now have a set of conditions and local strategies that prevent globally bad strategies from being played. In particular, we note that in every attacker-victim case from above, there is a set of hyperparameters that can be agreed on pre-payment that prevents the viability of a bad strategy given rational attackers. In fact, with proper hyperparameter selection, even irrational attackers cannot make reasonable attacks against the network, as all potential attacks in a proper hyperparameter set result in vanishingly small damage. In all profit-driven or benign actor cases, the incentives drive these actors to agree on hyperparameters that minimize the negative effects of randomness. Whether under benign actor assumptions or adversarial assumptions, the local incentives of each player naturally prevent bad strategies from being feasible.

6. Attacks and Defenses

In this section we evaluate specific use cases, summarize the main attacks relevant adversaries may employ and analyze our design's ability to defend against them.

6.1 Threat Model

We can partition the main goals of adversaries into several (non-exclusive) categories:

- *Traffic confirmation*: the adversary seeks to confirm whether user A is communicating with destination B, where A is some known user and B is some known destination entity (e.g. website).
- *Traffic analysis*: the adversary seeks to know all or some of the set of users A* who are communicating with destinations B*, along with associated metadata.
- *Traffic blocking*: The adversary seeks to block connections between some set of users A* and some set of destinations B*.
- *Content modification*: The adversary seeks to overtly or covertly modify the content of communication streams between some set of users A* and destinations B*.

We assume limited local active adversaries with some combination of powers:

- *Observation*: passively observe some fraction of network traffic
- *Infiltration*: control some fraction of Pzdid or Ethereum nodes or external servers
- *Manipulation*: actively modify some portion of network traffic
- *Inference*: apply compute on harvested data to infer *unobserved* information of interest

Pzdid can not protect against a stronger global adversary who can observe or modify all traffic or nodes. We assume an economic model where the Adversary's powers are practically limited by costs, most of which scale per user.

Traffic Analysis (Inference) Attacks

There is an extensive body of research concerning inference attacks on anonymity systems (and Tor in particular), which we can partition into a few main categories:

In *passive flow correlation*, the adversary observes traffic at one or more points on the network (typically at ingress and egress locations) then uses statistical inference to correlate streams through a multi-hop circuit[54,55][56,57]. Recent advances in deep learning increase the cost effectiveness of these attacks[54].

Using *active flow correlation*, the adversary can also manipulate traffic (e.g. insert timing delays) to create a watermark pattern to greatly boost precision and recall[58–60]. These attacks require control of hardware at the stream ingress to inject the traffic watermark.

Side channel correlation attacks are also possible in a low latency relay: timing measurements on one stream can still reveal sufficient information to correlate unobserved streams passing through the same relay[61][62]. These attacks can reveal likely nodes of the circuit but are generally insufficient to trace the complete circuit back to the user's IP.

Website fingerprinting attacks allow an adversary observing just the egress point of a connection to correlate streams through the circuit based on matching traffic patterns against a known library of website specific

fingerprints[63],[64]. Deep learning techniques can generate these fingerprints automatically[65–67]. Whether website fingerprinting attacks yet have sufficient precision/recall in the wild to be of practical use to Adversaries is debatable[68].

Scope

Given the wide space of possible adversary goals, capabilities and budgets, defending generically against all or a wide space of attackers is beyond the scope of a low latency, high bandwidth overlay network like Pzdid [26]. We will instead focus on some of the most common *economically relevant* use cases and their implied adversary models.

6.2 Bypassing Geographic Content Restrictions

Bypassing geographic restrictions on web content is one of the most common use cases for VPNs today[20]. Streaming services such as Netflix enforce geographic license restrictions by inferring a user's location from their IP address and then limit content access to a library customized for that specific location.

The adversary in this case has the goal of content modification and controls the destination website itself, which presents some interesting challenges. It is fairly easy for the adversary to simply detect most common VPN or proxy services by IP address and then block website access completely[21]. Using basic forms of target traffic analysis, the adversary can use IP registration databases to find IP address ranges associated with known VPN companies, or can look for a large number of different accounts sharing the same IP address to determine that a particular address is very likely that of a proxy or VPN server.

There are several strategies that current VPNs can use to provide clients with an obfuscated IP address suitable for evading geographic content locks. The simplest, but most expensive, is to provide individual clients with a unique IP address as an add-on service. Alternatively, VPNs can rapidly turnover IP addresses (through subleasing, etc) to provide a constant flow of fresh unblocked addresses for clients.

In principle Pzdid 's metadata registry (section 4.2) allows bandwidth sellers to advertise unique IP addresses using a custom tag (e.g. "unique_ip"). Clients could then filter on this tag along with geolocation to find exit nodes claiming to use a unique IP address in a specific location. The barrier to this is that the Pzdid market is built around the assumption of quick, stateless, semi-anonymous transactions whereas a unique IP address has a significant setup cost. A user who connects to a node actually offering a fresh unique IP address then disconnects a few seconds later will end up paying microdollars for a service that is roughly a million times more expensive to provide. Instead an Pzdid seller could charge a larger macropayment amount for a unique IP address service; this would require explicit user approval of a large invoice in the client UI and we expect would only reach feasibility for highly trusted curated sellers.

Alternatively, sellers may choose to directly advertise unblocking a specific streaming service. The implementation of this claimed capability is up to the seller: they could implement unblocking through rotation of fresh IP addresses and low user/IP address ratios. If successful, a seller could charge more for bandwidth with this service without requiring upfront macropayments.

In the long run Pzdid has a key advantage for this use case by allowing users access to servers from a variety of different providers, avoiding the lock-in risks inherent to the current VPN model. With a single VPN subscription

the user has little recourse when a particular provider's servers are suddenly blocked; with Pzdid the user can easily and near-instantly switch providers at any time.

6.3 Peer-to-Peer Sharing Systems

Peer-to-Peer networks are a popular means for users to share content directly, bypassing centralized content sources. ISPs (Internet Service Providers) may wish to limit or interfere with peer-to-peer sharing networks for various reasons: they may perceive them as threats to their cable television or streaming revenue, they can use large amounts of bandwidth, and they can allow users to share protected content. The adversary's goal is primarily one of deterrence which begins with traffic analysis: they wish to identify users who are using a particular p2p network and or sharing particular content.

The adversaries in this use case has fairly limited powers: their main attack strategies are to either detect and then shape or filter p2p packets or to infiltrate the peer-to-peer network by running their own nodes which then log IP addresses, actions and metadata of particular users. Current popular peer-to-peer networks such as Bittorrent have low economic security; infiltrating these networks is inexpensive. VPN's protect adequately for this use case in many jurisdictions by both encrypting traffic and simply hiding the user's IP address; this is feasible as long as the VPN is under no legal or financial obligation to keep logs or acquiring logs is difficult for the adversary.

Pzdid can provide a capable defense similar to that of VPNs for this use case through the combination of the stake-weighted selection mechanism and whitelists. An Pzdid client using a whitelist that includes only trusted providers known to avoid logging has a similar or better probability of avoiding node and adversary collusion as a user picking a VPN at random from a list of VPNs known to avoid logging.

The adversary succeeds with this attack when the user selects *both* an Pzdid node and a p2p file-sharing network (e.g. Bittorrent) node that the adversary controls. The probability this occurs is:

$$p(compromise(x,y) : x \in A_o, y \in A_B) = p(x \in A_o)\, p(y \in A_B) \quad (20)$$

$$p(y \in A_O) = \frac{S_{A \cap W}}{S_W} \quad (21)$$

$$p(y \in A_B) = \frac{B_A}{B_T} \quad (22)$$

x, y : the selected Pzdid node and file-sharing node, respectively
A_o, A_B : the set of Pzdid nodes and file-sharing nodes the adversary controls, respectively
W : the client's whitelist, a set of Pzdid nodes
S, S_W : the total OXT stake and OXT stake of nodes in W, respectively
$S_{A \cap W}$: the total OXT stake of nodes in $A \cap W$, the set of adversary nodes also in W
B_A, B_T : the adversary's bandwidth and the total bandwidth on the file-sharing network, respectively

Without a whitelist W, then S_W is equal to the total system stake S_T and the probability of selecting an adversarial Pzdid node is just $\frac{S_A}{S_T}$, the relative fraction of all OXT stake the adversary controls. In a hypothetical scenario where Pzdid has several million users and the total Pzdid stake value is around $1 billion (section 4.4), an adversary with a budget of $10 million for Pzdid nodes has a success rate that is three orders of magnitude lower for single-hop Pzdid users compared to unprotected users. For any Pzdid user connecting to a file-sharing node the adversary controls, the probability that user also connects to one of the adversary's Pzdid nodes is only 0.1%.

A random whitelist has no effect because in this case $\frac{S_{A \cap W}}{S_W} = \frac{S_A}{S}$. A carefully chosen whitelist reduces $S_{A \cap W}$ much more than S_W and can significantly reduce the compromise probability.

Assuming the adversary does not have the ability to execute effective traffic timing analysis attacks, a multi-hop circuit can significantly lower the selection probability:

$$p(compromise(X_k)) = \left(\frac{S_{A \cap W}}{S_W}\right)^{\lceil k/2 \rceil} \frac{B_A}{B_T} \qquad (23)$$

Here X_k represents a k-hop circuit; the adversary must control every other node in this circuit to infer the complete path. For a typical 3 hop circuit, the attacker must control 2 specific nodes: the first and last. Using the same parameters from above without a whitelist, the probability that a user connected to the adversary's file-sharing node is also using a compromised 3 hop circuit is now only 10^{-6}.

An advanced adversary could use active flow correlation analysis to reduce the effectiveness of multi-hop circuits. By injecting temporal fingerprint patterns into the traffic stream and detecting them at the endpoint, in theory an adversary could correlate and compromise even a lengthy circuit by only controlling the first Pzdid entrance node and the endpoint (the filesharing node in this case)[23-25]. The Pzdid client can help defend against these attacks through the optional use of *bandwidth burning*: padding the packet stream with dummy data packets to emulate a continuous low variance flow in an attempt to erase detectable temporal signals.

However, in this use case we believe these advanced traffic analysis attacks are unlikely. This type of Adversary has a very limited per user budget. Traffic analysis techniques provide statistical correlation evidence that is useful for surveillance but generally have significant false positive rates.

6.4 Avoiding ISP Censorship

Many countries now censor politically objectionable internet content[69], which is typically enforced by local ISPs (Internet Service Providers). The extent of censorship varies considerably from country to country but we can roughly divide this use case into two main categories: countries that censor but permit VPN use (e.g. Indonesia, Pakistan, Thailand), and more restrictive countries that censor extensively and also outlaw or restrict VPN usage (e.g. China, Russia).

Weak Censoring Adversaries

Using Pzdid to evade internet censorship in countries where VPN/proxy services are permitted is straightforward. The client could use a simple geographic filter to select from nodes outside the restricted country, but in practice this may be unnecessary because exit nodes in restricted countries are unlikely to receive much traffic anyway, and thus exit nodes will already tend to cluster in locations with little censorship. The adversary in these countries is 'weak' in the sense that it does not invest significant resources into preventing censorship evasion.

Strong Censoring Adversaries

The countries where VPN/proxy services are actively restricted present more of a challenge. China in particular has implemented an extensive technological solution for comprehensive internet surveillance and censorship, dubbed the Great Firewall of China(GFW). China has even begun issuing fines to individuals caught using VPNs[70]. Nonetheless, external VPNs remains popular in China[71], with providers playing a constant game of cat and mouse. This adversary has many capabilities, but three in particular are especially relevant for censorship evasion:

- The GFW uses *deep packet inspection* to detect likely VPN/proxy servers en masse.
- The GFW employs *active probing* to inspect suspected servers[72]
- The GFW uses automated and manual processes to ban IP addresses associated with VPN/proxy services

The Pzdid client builds tunnel connections using WebRTC which adds a layer of obfuscation to evade detection from deep packet inspection tools tuned for generic VPN/proxy recognition. However, if Pzdid becomes popular in China, it is likely that they will adapt the GFW packet inspection systems to recognize Pzdid WebRTC traffic, requiring further obfuscation plugin development.

More problematically, the main Pzdid discovery process relies on a public node directory published on the Ethereum blockchain (section 4.2). Once Pzdid is popular enough in China to warrant attention, it is quite likely that the GFW will automatically monitor the Ethereum blockchain and ban the IP addresses of all listd Pzdid nodes from the public directory.

Despite these obstacles, Chinese citizens could still use Pzdid as-is in a limited grass roots fashion where friends and enthusiasts outside the country run (potentially free) entry nodes and then share the addresses privately. Supporters and philanthropists could further support this cause by distributing OXT cryptocurrency along the same private social channels as the secret Pzdid node addresses. Core design improvements to better evade the GFW and facilitate OXT distribution into China are exciting future research directions (section 7).

6.5 Surveillance Evasion

Internet surveillance is generally more widespread than internet censorship. ISPs in most jurisdictions have some legal obligation to comply with valid surveillance requests from law enforcement, and the widespread extralegal surveillance operations of major western intelligence agencies is now an open secret. We will decompose this broad scenario into several models assuming different combinations of capabilities for the Adversary.

Passive ISP Monitoring

In much of the world, Internet Service Providers (ISPs) have the capability and affordance to monitor and log the internet traffic of their customers. In some jurisdictions logging is required by law to aid in law enforcement investigations. The ISP also may analyze packets for the purpose of traffic shaping to prioritize some applications over others for strategic reasons. They may collect and sell a user's browsing history to advertisers.

In these scenarios we assume the adversary lacks the motivation and capacity to infiltrate the Pzdid network and or destination endpoints. As long as the connection endpoint is not also under the ISP's control a single hop circuit suffices to evade generic untargeted traffic analysis surveillance in this case. The WebRTC encoding will also make Pzdid traffic look like regular web requests to cursory packet analysis tools, but will not fool an adversary who is familiar with Pzdid and uses more sophisticated deep packet inspection techniques.

A single-hop circuit provides less protection against an Adversary employing website fingerprinting techniques[65–67]. Multi-hop circuits reduce the precision/recall effectiveness of these attacks but not enough to render them ineffective. We assume that these correlation techniques are too expensive to employ *en masse*, but are a potential threat for targeted users.

Passive ISP and Endpoint Monitoring

In our next scenario the Adversary gains the ability to monitor endpoint traffic, but they still can not actively shape or control the user's entry traffic through their ISP. This scenario corresponds to an agency that is actively

surveilling specific endpoints (e.g. websites) and using traffic analysis to gather information on users of those targeted endpoints. Once the adversary finds IP addresses of targeted users, they next use that to acquire additional traffic logs and personal information about the user from their ISP.

The Adversary can now additionally employ passive flow correlation techniques[20-22], but again we assume that these techniques are too expensive to employ *en masse* across all traffic flowing through the ISP. Instead the Adversary has some limited analysis budget and must target likely user IP addresses for correlation.

A single hop circuit still suffices to evade surveillance in this case, assuming the endpoint connection is also encrypted using HTTPS/SSL and the user is not already targeted. The Adversary will only see a connection from the Pzdid node to the endpoint, but will not be able to easily determine the user's IP address.

As discussed in *section 5.8*, an Adversary fully monitoring traffic at the endpoint may be able to correlate the timing of traffic between the Pzdid node and the endpoint with the redemption of a winning ticket by that node. The ticket will reveal the payer's Pzdid nanopayment address, which the Adversary could then trace back to the user. Users can avoid this by taking appropriate steps to anonymize their OXT cryptocurrency.

Endpoint and Pzdid Infiltration

Now we consider an Adversary that crucially does not have the capability to monitor data at the user's ISP, but instead can infiltrate the endpoint and/or the Pzdid network. This model is realistic for users whose ISPs do not log traffic at scale or do not share significant traffic data with Adversaries. Flow correlation attacks are now much more difficult without the capability to monitor traffic on the link from the user to their first Pzdid node.

The adversary can infiltrate the Pzdid network to perform flow correlation attacks. The effectiveness gained through infiltration depends on the Adversary's budget for Pzdid nodes. The staking mechanism ensures a relatively high capture cost per user, and additionally as Pzdid gains users the cost of capturing a fixed percentage of Pzdid connections increases in proportion, as discussed in *section 4.4*. The Adversary can compromise the circuit by either requesting logs from a colluding Pzdid node operator that keeps traffic logs and provides them to the Adversary, or by controlling the Pzdid node directly. The compromise probability for a single node is:

$$p(compromise(x)) = p(x \subseteq \alpha) + (1 - p(x \subseteq \alpha))p(x \subseteq A) \quad (24)$$

$$p(x \subseteq \alpha) = \frac{S_{\alpha \cap W}}{S_W} \quad (25)$$

$$p(x \subseteq A) = \frac{S_{A \cap W}}{S_W} \quad (26)$$

x : the randomly selected Pzdid node
α : the set of colluding Pzdid nodes that log data for the Adversary
A : the set of Pzdid nodes the adversary controls directly
W : the client's whitelist, a set of Pzdid nodes
S_W : the total OXT stake of nodes in W
$S_{\alpha \cap W}$: the total OXT stake of nodes in $\alpha \cap W$, the set of colluding nodes also in W
$S_{A \cap W}$: the total OXT stake of nodes in $A \cap W$, the set of adversary nodes also in W

If the Adversary needs direct IP address metadata to confirm links, then for a multi-hop circuit they will need to compromise every edge and thus every other node. The multi-hop circuit compromise probability is thus a power function of the single-hop probability:

$$p(compromise(X_k)) = \left(\frac{S_{a \cap W}}{S_W} + \left(1 - \frac{S_{a \cap W}}{S_W}\right)\frac{S_{4 \cap W}}{S_W}\right)^{[k/2]} \quad (27)$$

Multi-hop circuits can provide significantly greater security, unless the Adversary can afford traffic analysis and imperfect statistical precision/recall is acceptable. If the Adversary uses flow correlation techniques as discussed in *section 6.1* then multi-hop circuits provide compromise probability more similar to single-hop circuits (eq. 24).

Strong Adversaries

More powerful Adversaries may have the ability to control packets at the ISP or AS (Autonomous System) level. Even an Adversary which only has the capability to monitor traffic at the user's ISP could still correlate users to websites through a multi-hop circuit using website fingerprinting attacks, the primary obstacle being cost. Clients can use *bandwidth burning* to provide a degree of protection against these attacks: padding the encrypted traffic stream to send uniform size packets on a highly regular schedule insensitive to the underlying data stream breaks the temporal correlations that most traffic analysis techniques depend on. Adversaries with significant per user analysis budgets and stronger sensing or inference capabilities could defeat multi-hop circuits, absent these additional protection measures. We discuss these possibilities as future work in the next section.

7. Future Work

Pzdid enables a bandwidth marketplace for decentralized proxy services through scalable off-chain nanopayments. Starting with this foundation, we have identified numerous routes for improvement in anonymity, usability, censorship resistance, and economic security.

Traffic Analysis Resistance

Pzdid 's current routing design minimizes latency and maximizes bandwidth at the expense of anonymity in the presence of traffic analysis attacks. These tradeoffs between latency, bandwidth, and anonymity are likely fundamental[26]. Users who desire stronger anonymity can use bandwidth burning (constant-rate transmission streams) that can help defeat traffic analysis by erasing most of the time-varying signature. Further improvements beyond bandwidth burning are likely required to defeat various inference attacks[73], and we leave a full analysis to future work. Independent improvements to latency aware route construction could enable longer circuits at the same latency, and improved mixing by using a sparser connection graph with more streams mixed along fewer active edges.

Payment Anonymity

Pzdid 's nanopayment system is built on Ethereum and thus is only semi-anonymous. Users requiring full payment anonymity thus need to externally anonymize their OXT cryptocurrency before funding nanopayment accounts, which creates a usability hurdle.

Alternatively, Pzdid nanopayments and circuits themselves could allow high speed mixing. The directory service could be repurposed to advertise nodes that provide mixing and/or register mixing peers. This use case could potentially strain the double-spend and griefing defense mechanisms (5.10), so may require improvements to double-spend detection and prevention.

Low-Variance Nanopayments

The current Pzdid nanopayment mechanism has a fundamental variance/overhead tradeoff. The core source of variance is the statistical independence of tickets. The variance could potentially be eliminated by using a mutually exclusive ticket scheme. In simplest form this could entail a single winning ticket per payment account. As there is only a single winner for an entire set of tickets, the variance is eliminated. One tradeoff is that mutually exclusive tickets would require deferring ticket winner determination into the future, using a multi-party source of entropy instead of a simple two party entropy protocol. The ethereum blockchain itself can be used as a simple source of entropy and is probably sufficiently secure for the small transaction values nanopayment settlements require. However, deferred winner determination entails a much larger volume of unsettled payments in flight, incurring additional per nanopayment storage costs.

Traffic Obfuscation

There is an ongoing arms race in the competing research fields of traffic obfuscation and detection. Traffic obfuscators use strategies such as randomization[74,75], transformation/mimicry[76], tunneling[76,77], and generative modeling[78]. Unfortunately all of these techniques are susceptible to machine learning based detection[27] systems trained on examples of real and obfuscated traffic. Generally stronger obfuscators require more compute per byte. The obfuscation problem can be formulated as a type of GAN[79] objective where the generator learns to transform a traffic stream to evade detection while preserving a reversibility or reconstruction property, and the discriminator learns to distinguish between real and transformed streams. This opens the door for deep learning based obfuscators (and detectors).

Improved Censorship Resistance

Pzdid ’s ability to evade state level censorship is primarily limited by the public advertising of nodes on the Ethereum blockchain. Stronger censorship resistance will require some form of private advertising. We can model this as a game where a bandwidth seller seeks to advertise unblocked IP addresses to legitimate customers while hiding them from the Adversary. The seller gains some expected future revenue value for every legitimate customer that learns of the IP address, but once the Adversary discovers the IP address and blocks it any remaining future revenue value is lost. A viable strategy for the seller is to use an affiliate scheme to reward advertising peers with a fraction of the future revenue stream. This will create a market niche for affiliates who are good at finding and advertising node addresses to legitimate users while avoiding adversarial colluders.

Whitelist Surety Bonds

We could magnify the positive incentive alignment affects of staking and stake-weighting by allowing OXT to be staked on a node’s inclusion in a particular whitelist. If the node is ever removed from this list (before the stake is withdrawn), then the stake deposit would be forfeited and burnt. This stake would become something like a surety bond, allowing node providers to prove trustworthiness by putting their money at risk in the event of bad behavior. The idea is simple but requires careful incentive design and verification.

We welcome you to develop your own curated lists with innovative incentive structures.

8. Acknowledgements

Pzdid has been a collaborative team project and we would like to especially thank Gustav Simonsson and David Salamon for their substantive intellectual contributions (including authorship of version 0.9.2 of the whitepaper).

References

1. Dingledine R, Mathewson N, Syverson P. Tor: The Second-Generation Onion Router [Internet]. 2004. Available from: http://dx.doi.org/10.21236/ada465464

2. Shahbar K, Nur Zincir-Heywood A. Effects of Shared Bandwidth on Anonymity of the I2P Network Users [Internet]. 2017 IEEE Security and Privacy Workshops (SPW). 2017. Available from: http://dx.doi.org/10.1109/spw.2017.19

3. Chaum D. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms [Internet]. Advances in Information Security. 2003. p. 211–9. Available from: http://dx.doi.org/10.1007/978-1-4615-0239-5_14

4. HashCash [Internet]. 2002 [cited 2019 Sep 10]. Available from: http://www.hashcash.org/hashcash.pdf

5. Bitcoin: A Peer-to-Peer Electronic Cash System [Internet]. [cited 2019 Sep 10]. Available from: https://bitcoin.org/bitcoin.pdf

6. Pzdid 0.9.2 [Internet]. 2019 [cited 2019 Sep 10]. Available from: https://www.Pzdid .com/assets/whitepaper/whitepaper.pdf

7. Stoica I, Morris R, Liben-Nowell D, Karger DR, Kaashoek MF, Dabek F, et al. Chord: a scalable peer-to-peer lookup protocol for internet applications [Internet]. Vol. 11, IEEE/ACM Transactions on Networking. 2003. p. 17–32. Available from: http://dx.doi.org/10.1109/tnet.2002.808407

8. Wood DD. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. 2014 [cited 2019 Sep 11]; Available from: https://pdfs.semanticscholar.org/ee5f/d86e5210b2b59f932a131fda164f030f915e.pdf

9. A Protocol for Packet Network Intercommunication [Internet]. The Best of the Best. 2009. Available from: http://dx.doi.org/10.1109/9780470546543.ch54

10. Fadilpa&scaron S, i&#. China “hijacked traffic” to spy on the West [Internet]. ITProPortal. ITProPortal; 2018 [cited 2019 Nov 17]. Available from: https://www.itproportal.com/news/china-eavesdropping-on-western-communication-for-years-research-claims/

11. Bloomberg - Are you a robot? [Internet]. [cited 2019 Nov 17]. Available from: https://www.bloomberg.com/news/articles/2018-09-04/youtube-and-netflix-throttled-by-carriers-research-finds

12. Morran BC. House Votes To Allow Internet Service Providers To Sell, Share Your Personal Information [Internet]. Consumer Reports. [cited 2019 Nov 17]. Available from: https://www.consumerreports.org/consumerist/house-votes-to-allow-internet-service-providers-to-sell-share-your-personal-information/

13. Net Neutrality: Caught in a web of lobbying and regulatory uncertainty [Internet]. Sustainalytics. 2018 [cited 2019 Nov 17]. Available from: https://www.sustainalytics.com/esg-blog/net-neutrality-caught-in-a-web-of-lobbying-and-regulatory-uncertainty/

14. Rosenberg S. Facebook’s reputation takes a hit in new survey [Internet]. Axios. 2019 [cited 2019 Nov 17]. Available from: https://www.axios.com/facebook-reputation-drops-axios-harris-poll-0d6c406a-4c2e-463a-af98-1748d3e0ab9a.html

15. Marks G. Facebook Usage Drops 26 Percent…And Other Small Business Tech News This Week [Internet].

Forbes. Forbes; 2019 [cited 2019 Nov 17]. Available from: https://www.forbes.com/sites/quickerbettertech/2019/10/27/facebook-usage-drops-26-percentand-other-small-business-tech-news-this-week/

16. Brodkin J. 50 million US homes have only one 25Mbps Internet provider or none at all [Internet]. Ars Technica. 2017 [cited 2019 Nov 17]. Available from: https://arstechnica.com/information-technology/2017/06/50-million-us-homes-have-only-one-25mbps-internet-provider-or-none-at-all/

17. SSH Celebrates 20 Years as Industry Standard | SSH.COM [Internet]. [cited 2019 Nov 17]. Available from: https://www.ssh.com/press-releases/111-ssh-communications-security-celebrates-20-years-as-industry-standard

18. Fu X, Graham B, Bettati R, Zhao W. Active traffic analysis attacks and countermeasures [Internet]. 2003 International Conference on Computer Networks and Mobile Computing, 2003. ICCNMC 2003. Available from: http://dx.doi.org/10.1109/iccnmc.2003.1243024

19. Dixon C, Bragin T, Krishnamurthy A, Anderson T. Tit-for-Tat Distributed Resource Allocation. [cited 2019 Sep 23]; Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.182.1544

20. Karakaya M, Korpeoglu I, Ulusoy Ö. Free Riding in Peer-to-Peer Networks [Internet]. Vol. 13, IEEE Internet Computing. 2009. p. 92–8. Available from: http://dx.doi.org/10.1109/mic.2009.33

21. Ngan T-W "johnny," Dingledine R, Wallach DS. Building Incentives into Tor [Internet]. Financial Cryptography and Data Security. 2010. p. 238–56. Available from: http://dx.doi.org/10.1007/978-3-642-14577-3_19

22. Androulaki E, Raykova M, Srivatsan S, Stavrou A, Bellovin SM. PAR: Payment for Anonymous Routing [Internet]. Privacy Enhancing Technologies. p. 219–36. Available from: http://dx.doi.org/10.1007/978-3-540-70630-4_14

23. Ghosh M, Richardson M, Ford B, Jansen R. A TorPath to TorCoin: Proof-of-Bandwidth Altcoins for Compensating Relays. 2014 Jul 18 [cited 2019 Sep 23]; Available from: https://apps.dtic.mil/dtic/tr/fulltext/u2/a621867.pdf

24. A Protocol for Interledger Payments [Internet]. [cited 2019 Sep 23]. Available from: https://pdfs.semanticscholar.org/ab98/c62a7efdc5362c7f36589680597a93f3111f.pdf

25. Khosla A, Saran V, Zoghb N. Techniques for Privacy Over the Interledger. 2018 [cited 2019 Sep 23]; Available from: https://pdfs.semanticscholar.org/02f3/aae499723063cf9c3cc42508cae13d16aa7d.pdf

26. Das D, Meiser S, Mohammadi E, Kate A. Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency - Choose Two [Internet]. 2018 IEEE Symposium on Security and Privacy (SP). 2018. Available from: http://dx.doi.org/10.1109/sp.2018.00011

27. Wang L, Dyer KP, Akella A, Ristenpart T, Shrimpton T. Seeing through Network-Protocol Obfuscation [Internet]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15. 2015. Available from: http://dx.doi.org/10.1145/2810103.2813715

28. Budish E. The Economic Limits of Bitcoin and the Blockchain [Internet]. 2018. Available from: http://dx.doi.org/10.3386/w24717

29. Website [Internet]. [cited 2019 Oct 2]. Available from: https://bitinfocharts.com/comparison/ethereum-transactionfees.html

30. Bitcoin Avg. Transaction Fee chart [Internet]. BitInfoCharts. [cited 2019 Oct 2]. Available from: https://bitinfocharts.com/

31. Khattak S, Elahi T, Simon L, Swanson CM, Murdoch SJ, Goldberg I. SoK: Making Sense of Censorship Resistance Systems [Internet]. Vol. 2016, Proceedings on Privacy Enhancing Technologies. 2016. p. 37–61. Available from: http://dx.doi.org/10.1515/popets-2016-0028

32. Contributors to Wikimedia projects. ISO/IEC 7816 - Wikipedia [Internet]. Wikimedia Foundation, Inc. 2002 [cited 2019 Oct 2]. Available from: https://en.wikipedia.org/wiki/ISO/IEC_7816

33. EBICS.ORG: Home Page [Internet]. [cited 2019 Oct 2]. Available from: http://www.ebics.org/home-page

34. Website [Internet]. [cited 2019 Oct 2]. Available from: https://www.swift.com/

35. Website [Internet]. [cited 2019 Oct 2]. Available from: https://www.swift.com/

36. Website [Internet]. [cited 2019 Oct 2]. Available from: http://www.nyce.net/about

37. Website [Internet]. [cited 2019 Oct 2]. Available from: http://www.investopedia.com/terms/r/reconciliation.asp

38. [No title] [Internet]. [cited 2019 Oct 2]. Available from: https://www.aba.com/-/media/archives/endorsed/rippleshot-state-of-card-fraud.pdf

39. Mian A, Hameed A, Khayyam M, Ahmed F, Beraldi R. Enhancing communication adaptability between payment card processing networks [Internet]. Vol. 53, IEEE Communications Magazine. 2015. p. 58–64. Available from: http://dx.doi.org/10.1109/mcom.2015.7060519

40. Banks and WikiLeaks. NY Times [Internet]. 2010 Dec 25 [cited 2019 Oct 2]; Available from: https://www.nytimes.com/2010/12/26/opinion/26sun3.html

41. What are common credit card processing fees? [Internet]. Quora. [cited 2019 Oct 2]. Available from: https://www.quora.com/What-are-common-credit-card-processing-fees

42. Website [Internet]. [cited 2019 Oct 2]. Available from: https://www.nerdwallet.com/blog/banking/wire-transfers-what-banks-charge

43. Website [Internet]. [cited 2019 Oct 2]. Available from: https://www.valuepenguin.com/what-credit-card-processing-fees-costs

44. Website [Internet]. [cited 2019 Oct 2]. Available from: https://www.economist.com/blogs/dailychart/2010/12/remittances

45. Website [Internet]. [cited 2019 Oct 2]. Available from: https://financefeeds.com/alipay-vs-wechat-pay-vs-unionpay-important-research/

46. Joseph Poon TD. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments [Internet]. Available from: https://lightning.network/lightning-network-paper.pdf

47. [No title] [Internet]. [cited 2019 Oct 2]. Available from: https://www.usenix.org/system/files/conference/usenixsecurity15/sec15-paper-heilman.pdf

48. Wheeler D. Transactions using bets [Internet]. Security Protocols. 1997. p. 89–92. Available from: http://dx.doi.org/10.1007/3-540-62494-5_7

49. Rivest RL. Peppercoin Micropayments [Internet]. Financial Cryptography. 2004. p. 2–8. Available from: http://dx.doi.org/10.1007/978-3-540-27809-2_2

50. Pass R, Shelat A. Micropayments for Decentralized Currencies [Internet]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security - CCS '15. 2015. Available from:

http://dx.doi.org/10.1145/2810103.2813713

51. Ethereum Avg. Transaction Fee chart [Internet]. BitInfoCharts. [cited 2019 Oct 2]. Available from: https://bitinfocharts.com/

52. Website [Internet]. [cited 2019 Oct 2]. Available from: https://etherscan.io/chart/gaslimit

53. Website [Internet]. [cited 2019 Oct 2]. Available from: https://etherscan.io/chart/blocktime

54. Nasr M, Bahramali A, Houmansadr A. DeepCorr: Strong Flow Correlation Attacks on Tor Using Deep Learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM; 2018. p. 1962–76.

55. Borisov N, Danezis G, Mittal P, Tabriz P. Denial of service or denial of security? In: Proceedings of the 14th ACM conference on Computer and communications security. ACM; 2007. p. 92–102.

56. Sun Y, Edmundson A, Vanbever L, Li O, Rexford J, Chiang M, et al. RAPTOR: Routing Attacks on Privacy in Tor. 2015 [cited 2019 Sep 16]; Available from: https://pdfs.semanticscholar.org/76c7/73bb98b0a266970a589f2cabbd24565b6e19.pdf

57. Johnson A, Wacek C, Jansen R, Sherr M, Syverson P. Users get routed [Internet]. Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13. 2013. Available from: http://dx.doi.org/10.1145/2508859.2516651

58. Houmansadr A, Kiyavash N, Borisov N. Multi-flow attack resistant watermarks for network flows [Internet]. 2009 IEEE International Conference on Acoustics, Speech and Signal Processing. 2009. Available from: http://dx.doi.org/10.1109/icassp.2009.4959879

59. Zhang L, Wang Z, Xu J, Wang Q. Multi-flow Attack Resistant Interval-Based Watermarks for Tracing Multiple Network Flows [Internet]. Computing and Intelligent Systems. 2011. p. 166–73. Available from: http://dx.doi.org/10.1007/978-3-642-24010-2_23

60. Yu W, Fu X, Graham S, Xuan D, Zhao W. DSSS-Based Flow Marking Technique for Invisible Traceback [Internet]. 2007 IEEE Symposium on Security and Privacy (SP '07). 2007. Available from: http://dx.doi.org/10.1109/sp.2007.14

61. Murdoch SJ, Danezis G. Low-Cost Traffic Analysis of Tor [Internet]. 2005 IEEE Symposium on Security and Privacy (S&P'05). Available from: http://dx.doi.org/10.1109/sp.2005.12

62. Chakravarty S, Stavrou A, Keromytis AD. Traffic Analysis against Low-Latency Anonymity Networks Using Available Bandwidth Estimation. In: Computer Security – ESORICS 2010. Springer, Berlin, Heidelberg; 2010. p. 249–67.

63. Panchenko A, Niessen L, Zinnen A, Engel T. Website fingerprinting in onion routing based anonymization networks [Internet]. Proceedings of the 10th annual ACM workshop on Privacy in the electronic society - WPES '11. 2011. Available from: http://dx.doi.org/10.1145/2046556.2046570

64. Cai X, Zhang XC, Joshi B, Johnson R. Touching from a distance [Internet]. Proceedings of the 2012 ACM conference on Computer and communications security - CCS '12. 2012. Available from: http://dx.doi.org/10.1145/2382196.2382260

65. Rimmer V, Preuveneers D, Juarez M, Van Goethem T, Joosen W. Automated Website Fingerprinting through Deep Learning [Internet]. Proceedings 2018 Network and Distributed System Security Symposium. 2018. Available from: http://dx.doi.org/10.14722/ndss.2018.23105

66. Bhat S, Lu D, Kwon A, Devadas S. Var-CNN: A Data-Efficient Website Fingerprinting Attack Based on Deep Learning [Internet]. Vol. 2019, Proceedings on Privacy Enhancing Technologies. 2019. p. 292–310. Available

from: http://dx.doi.org/10.2478/popets-2019-0070

67. Sirinam P, Imani M, Juarez M, Wright M. Deep Fingerprinting: Undermining Website Fingerprinting Defenses with Deep Learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. ACM; 2018. p. 1928–43.

68. A Critique of Website Traffic Fingerprinting Attacks | Tor Blog [Internet]. [cited 2019 Sep 17]. Available from: https://blog.torproject.org/critique-website-traffic-fingerprinting-attacks

69. Pearce P, Ensafi R, Li F, Feamster N, Paxson V. Augur: Internet-Wide Detection of Connectivity Disruptions [Internet]. 2017 IEEE Symposium on Security and Privacy (SP). 2017. Available from: http://dx.doi.org/10.1109/sp.2017.55

70. Humphries M. China Starts Issuing $145 Fines for Using a VPN [Internet]. PCMAG. 2019 [cited 2019 Sep 15]. Available from: https://www.pcmag.com/news/365860/china-starts-issuing-145-fines-for-using-a-vpn

71. VPN Usage Statistics | Global Trends in the VPN Industry [Internet]. GeoSurf. 2019 [cited 2019 Sep 15]. Available from: https://www.geosurf.com/blog/vpn-usage-statistics/

72. Ensafi R, Fifield D, Winter P, Feamster N, Weaver N, Paxson V. Examining How the Great Firewall Discovers Hidden Circumvention Servers [Internet]. Proceedings of the 2015 ACM Conference on Internet Measurement Conference - IMC '15. 2015. Available from: http://dx.doi.org/10.1145/2815675.2815690

73. Chen, Chen C, Asoni DE, Perrig A, Barrera D, Danezis G, et al. TARANET: Traffic-Analysis Resistant Anonymity at the Network Layer [Internet]. 2018 IEEE European Symposium on Security and Privacy (EuroS&P). 2018. Available from: http://dx.doi.org/10.1109/eurosp.2018.00018

74. Meiklejohn S, Mercer R. Möbius: Trustless Tumbling for Transaction Privacy [Internet]. Vol. 2018, Proceedings on Privacy Enhancing Technologies. 2018. p. 105–21. Available from: http://dx.doi.org/10.1515/popets-2018-0015

75. Winter P, Pulls T, Fuss J. ScrambleSuit: A Polymorph Network Protocol to Circumvent Censorship [Internet]. 2013 [cited 2019 Sep 18]. Available from: http://arxiv.org/abs/1305.3199

76. Moghaddam HM. Skypemorph: Protocol Obfuscation for Censorship Resistance. 2013. 54 p.

77. Brubaker C, Houmansadr A, Shmatikov V. CloudTransport: Using Cloud Storage for Censorship-Resistant Networking [Internet]. Privacy Enhancing Technologies. 2014. p. 1–20. Available from: http://dx.doi.org/10.1007/978-3-319-08506-7_1

78. Dyer KP, Coull SE, Shrimpton T. Marionette: A Programmable Network Traffic Obfuscation System. In: 24th {USENIX} Security Symposium ({USENIX} Security 15). 2015. p. 367–82.

79. Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. In: Advances in Neural Information Processing Systems. 2014. p. 2672–80.