SAND2018-13059C

Monetary Tor Incentives with Efficient Nanopayment Channels

***Abstract*—Tor, the most widely used and well-studied traffic anonymization network in the world, suffers from orthogonal limitations in network diversity and performance. We propose to mitigate both problems simultaneously through the introduction of a premium bandwidth market between clients and relays. To this end, we present moneTor: a relay incentivization scheme featuring anonymous, efficient, and economically robust nanopayments. Our approach leverages the latest advances in cryptocurrency research toward a design that is directly integrated into the existing Tor architecture. This work describes a full-stack strategy that covers economic policy, novel payment algorithms, and networking implementation. Through live empirical data collection and analysis of our emulated prototype, we argue that moneTor is the first technically feasible monetary incentive scheme for Tor, offering upwards of 100% improvements in differentiated bandwidth for paying users at near-optimal throughput and latency overhead.**

***Index Terms*—Tor, cryptocurrency, payment channels**

I. Introduction

Anonymous traffic routing through Tor remains one of the most popular low-latency methods for censorship evasion and privacy protection [1]. In this setup, clients protect their TCP/IP metadata and content by routing their traffic through an onion encrypted path with three randomly selected volunteer relay nodes, referred to as a circuit. The network presently consists of $\approx 6,400$ relays contributing over 230 Gbit/s of bandwidth globally [2]. While Tor has proven to be a highly effective option for privacy-seeking users, it suffers from two orthogonal issues that are relevant to this work. The first is the broad family of collusion attacks. These threats are relevant in scenarios where an attacker, who controls multiple nodes or network vantage points, is probabilistically placed in key roles along a single circuit, opening a much easier path to client deanonymization [3], [4]. The second problem is performance. Although the overlay protocol itself generates an inherent overhead in network resources, Tor suffers from additional traffic congestion that leads to suboptimal network performance [2], [5].

One approach to mitigate these problems is to address them separately from a networking standpoint. Indeed, a significant portion of recent research in Tor proposes modifications to the core protocol itself, such as reengineering the TCP+TLS part of the stack [6] or designing a better kernel-aware scheduler [7]. A second approach observes that the anonymity and performance of the network are both proportional to the number of nodes and users. From this perspective, the problem becomes a largely economic question: how can we incentivize more relay participation? There is a long line of research that explores various strategies for incentivization spanning over the past decade. Progress in this field faces a multitude of challenges. Consider the approach most relevant to our work: monetary incentives. Aside from the analytically out-of-scope set of legal and sociopolitical obstacles, monetary payments in this environment must overcome a trifecta of challenging constraints:

- *Anonymity*: The paramount mission of Tor is user privacy. This cannot be compromised or reduced by transparent money transaction trails.
- *Payment Security*: Financial transactions within an anonymity system cannot transpire without strong guarantees of cryptographic security.
- *Efficiency*: Tor services millions of concurrent users, all of whom maintain largely short-term relationships. A robust payment system must handle extremely lightweight and scalable payments so as to accommodate the dynamic and often short-lived activities of these clients.

Present Landscape While a number of prior works satisfy some subset of these constraints, no single proposal has thus far been sufficiently convincing so as to warrant further development. We speculate that the lack of a breakthrough in this niche area is not a matter of insufficient ingenuity but rather one of timing. In recent years, there has been an explosion of academic research within the domain of cryptocurrencies. As of this writing, Satoshi Nakamoto's Bitcoin whitepaper has already garnered over 4,000 references in citing publications [8]. [1] This body of work has sparked the development of a multitude of new techniques that will prove indispensable to our own area of research. Our objective then, is to leverage the full extent of these innovations into a practical next-generation incentivization strategy for Tor.

Contributions The moneTor scheme is a novel full-stack framework for Tor incentives. We make the following contributions in this paper:

1) Discuss economic considerations for the Tor network to support Tor as a public good;
2) Introduce new highly-efficient nanopayment protocols which satisfy all *Anonymity*, *Payment Security* and *Efficiency* constraints. These protocols are analyzed and may be of some independent interest for other high-frequency payment applications as well;
3) Detail our integration strategy to add a payment layer which extends the existing Tor architecture;
4) Collect privacy-preserving client usage data in order to justify design parameters and argue for their efficacy;
5) Identify an efficient method of throttling to give priority users an advantage;

[1] The recency of this development is highlighted by the fact that about half of this work is dated after 2017

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

6) Implement a prototype of our nanopayment protocols extending the Tor protocol in the core C software, featuring more than 15k lines of code;
7) Conduct large scale simulations to analyze the performance impact of the embedded payment and incentivization scheme.

The moneTor design leverages fully-specified algorithms for some components and well-studied existing research for all others. On a technical level, we believe that moneTor could be feasibly developed today within a deployed system.

Roadmap.: In Section II, we draw on technical preliminaries from two distinct fields: applied Tor research and cryptographic payment channels. Section III presents the related work. Our contributions begin formally with Section IV, where we first describe a high-level economic model for the flow of money through the network. In Section V, we outline the technical construction of our nanopayment scheme at the payment protocol level. Section VI expands the technical construction of moneTor to cover modifications at the network level. In section VII we justify our design decisions with real-world data collection from live Tor users. We continue in Section VIII with a validation of our technical design, carried out through experiments performed on a native proof-of-concept implementation. Finally, we discuss limitations and future work in Section IX, reference our source code in Section X, and present concluding remarks in Section XI.

II. Background

A. Tor

1) Tor Architecture: The Tor network is composed of multiple different components, each of which run the same code base. Volunteers can run relays and enable specific roles or tasks such as *network consensus directories*, *HSDirs* and *Exit policies*. *Directory authorities* and *Bandwidth authorities* are the most important components of the network, and are only operated by trustworthy core contributors. There are currently nine directory authorities which periodically reach agreement over the state of the network, called the *consensus document*. This consensus document holds the identification information related to all available relays inside the network, as well as the result of the authority vote (e.g., a set of *flags* associated to each relay). The bandwidth authorities constantly measure every relay and provide to the directory authorities a measurement value for each of them, which plays a critical role in the path selection algorithm. This paper extends the architecture by adding two new roles needed for our monetary relay incentivization: $intermediary$ and $ledger$.

2) Traffic Analysis: Tor's threat model assumes a local adversary who can observe some fraction of the network and can operate or compromise a number of onion routers. Tor also assumes a local adversary who can manipulate user streams by inserting, modifying, deleting or delaying data to create observable perturbations. Typically, by observing both ends of an anonymous stream, an attacker can infer the participating parties using statistical correlation. The adversary's precision can be further improved by adding traffic flow perturbations. This attack is called *end-to-end correlation*. Tor does not implement explicit countermeasures to this attack but does strive to minimize its impact. However, Rochet and Pereira recently showed that Tor's essential forward-compatibility feature can be exploited in a silent near-perfect and instantaneous active traffic confirmation attack [9]. These developments imply a serious need to induce more diversity to the Tor network, which would make such traffic analysis against a large fraction of Tor users more costly. One such solution is presented in this paper.

3) Circuit handling on Tor clients: Application traffic in Tor is handled through streams which are multiplexed with other streams within a circuit. In order to reduce latency in the user experience, Tor attempts to build circuits preemptively as soon as the client obtains the necessary directory information. Once a stream is created by a user application, it can be immediately routed through the idle circuit, dramatically improving latency. The anticipated number of required preemptive circuits is dynamically calculated every second by the Tor client. We exploit this same strategy for moneTor's circuit setup routine.

4) Evaluating Tor's performance: Shadow [10] is a discrete event networking simulator that allows real, unmodified applications to run within a virtual network. Its primary advantage is the ability to run native networking application code that interfaces with the simulator via an external application-specific plugin. Within the simulation environment itself, Shadow faithfully mimics the real Tor network conditions including bandwidth and latency. As a result, experiments conducted with this tool tends to be more accurate than ones conducted over alternatives such as private university networks or PlanetLab [11]. We utilize the shadow framework to evaluate the costs and performance of our algorithms over network-level simulations.

5) Flow Control: Tor seeks to maintain an optimal flow of cells in each circuit by capping the rate of transmission to fixed-sized windows. The CIRCWINDOWSTART value dictates flow for the overall circuit while STREAMWINDOWSTART dictates each end-to-end flow between a client and the edge connection at an exit relay.[2] The number of cells going in each direction are tracked separately. In effect, Tor attempts to keep the flow of cells on each circuit as full as possible while ensuring that the number of cells in transition do not exceed the limit. Flow control has a strong impact on network conditions as shown by AlSabah *et al.* [12] in work toward improving total global performance. In the design of moneTor, we will adapt these flow control windows toward our somewhat different objective to prioritize paid traffic.

B. Payment Channels

1) Payment Channels: The core cryptocurrency component featured in moneTor is the tripartite bidirectional micropayment channel. Base layer cryptocurrency protocols are typically capped on the order of tens of transactions per

[2]Default values, expressed in number of cells, are CIRCWINDOWSTART = 1000 and STREAMWINDOWSTART = 500

seconds [13]. The most actively pursued path toward better scalability thus far is work in off-chain payment channel networks [14]. In this setup, a single ledger transaction is used to escrow funds by two parties A and B. These parties may then proceed to make bidirectional micropayments to each other *without ledger interaction* through the exchange of signed "I Owe You" tokens. By themselves, channels are useful for reducing the number of ledger interactions for parties with reoccurring interactions. More consequential for the scalability problem is the tripartite channel paradigm in which A pays B through some *intermediary* party I with which they both maintain active channels. In practice, A and B might occupy the roles of a customer and merchant who are registered with a well-known financial service provider I. A, B, and I need only interact with the ledger periodically to deposit and withdraw large sums of money, improving the network capacity by multiple orders of magnitude. Informally, tripartite channels are secure if the following requirements are met.

1) At every step of the protocol, all parties possess proof of execution of the last finalized payment state
2) Given two proofs of payment state, the network can unambiguously identify the more recent state.
3) When A agrees to pay B through I, the payment is atomic. That is, there is never a situation in which I pays B but is unable to extract the agreed-upon payment from A.

The guarantees are achieved in the Bitcoin Lightning network and other similar protocols through simple hash commitment and transaction delay primitives. The end result is a game-theoretic notion of security whereby all parties are incentivized to behave honestly.

2) Anonymous Payment Channels: The micropayment channel concept has since been extended to support anonymity design goals. Z-Channel specifies an implementation designed specifically for Zerocash which only supports two-party channels [15]. Green and Miers designed Bolt, which is the only known anonymous micropayment scheme to date that supports three-party intermediary channels [16]. In this framework, the anonymity set is defined with respect to the collection of users connected to the same intermediary. In other words, given a set of end users $E_{all} = \{E_1, E_2, ...E_n\}$ who each have an active channel with I, E_a should be able to send a secure payment to E_b such that I cannot identify E_a or E_b from E_{all} nor can I infer the payment value. Of course, I must still be able to verify that the payment is valid and that its internal channel states have been updated accordingly. [3] Several nuances arise concerning end user privacy in certain situations, such as during the micropayment setup escrow phase and in the event that I maliciously aborts. We do not consider it necessary to discuss these caveats here as they are not prohibitively problematic for our later treatment of anonymous payment channels.

[3]This is achieved using a combination of zero-knowledge proofs and blind signatures, in the case of Bolt

III. Related Work

We classify incentivization schemes into three types of strategies: non-transferable benefits, transferable benefits, and monetary payments.[4]

A. Non-Transferable Benefits

These proposals aim to recruit relays by offering some privileged status intended for personal use which cannot be securely sold for financial profit.

1) Gold Star: As one of the earliest incentive proposals, Gold Star introduces the notion of premium bandwidth. Premium, or *gold star* status, is awarded exclusively to the fastest 7/8 fraction of relays. While highly attractive for its conceptual simplicity, Gold Star leaks a considerable amount of user information as the set of relays in the Tor network is now much smaller than the set of users [17].

2) BRAIDS/LIRA: The BRAIDS scheme introduces *tickets* to represent premium status. Small numbers of ephemeral tickets are freely distributed by a central *bank* to any client upon request or to relays that have accumulated spent tickets from clients. Crucially, tickets can only be spent at a single relay defined at the time of their minting to circumvent the double spending problem [18]. LIRA is an ideological successor to BRAIDs which reduces scalability problems imposed by the centralized infrastructure. Clients in LIRA probabilistically "win" premium tickets without any interaction with the bank. While LIRA improves the efficiency of BRAIDS, neither of these designs were constructed to prevent strong threats of client cheating [19].

moneTor has the advantage to both prevent any client cheating (i.e., they cannot obtain priority without paying for it) and to prevent double-spending without leaking timing information.

B. Transferable Benefits

The next class of transferable benefits describes schemes which offers privileges or products intended to hold value in a secondary resale market. These indirect financial incentives presume to attract a broader demand than in the non-transferable case.

1) TEARS: TEARS introduces a two-layer approach whereby *shallots* token are awarded by a central bank to participating relays. These shallots, which are securely exchanged tokens, can then be redeemed for BRAIDS-style *priority passes*. While fully exchangeable shallots represent an economic improvement over non-transferable privileges, these tokens are conceptually discrete and indivisible assets that are not as easily exchanged as true currency.

moneTor has the advantage to offer transferability with divisibility, and like TEARS, moneTor does not change the

[4]Generally speaking, incentives provided by monetary payments are more economically robust than transferable benefits, which in turn are more robust than non-transferable benefits. Although we examine related works through the prism of economic appeal, this is not a statement of superiority so much as an observation on diverging motivation. Indeed, a number of sociological and legal implications must be weighed before introducing financial profitability into the Tor ecosystem. We consider these to be out of scope for the purposes of our research.

base of the Tor design and can be integrated inside the current architecture.

2) TorPath to TorCoin: TorCoin proposes to revamp the Tor architecture into one which can serve as the basis for a new cryptocurrency mined with proof-of-bandwidth. The networking component specifies verifiable pseudorandom shuffling as the new method for user circuit selection. This modified protocol would in theory provide a weakly secure means for relays to mint new TorCoin tokens based on their bandwidth contribution [20]. While researching on TorCoin-like approach has the potential to help the Tor project on one of the fundamental security problems (i.e., the bandwidth measurement system), outcomes are not clear as many vulnerabilities and deployment questions remain.

3) Proof-of-work as anonymous micropayment: Users in this simple design provide partial proof-of-work tokens that can then be redeemed by relays for profit in real cryptocurrency mining pools. The drawback of this scheme is simply an issue of unrealistic magnitude; the paper estimates that a user who continuously mines with typical consumer CPU hardware will be able to make only a few cents worth of payments every 24 hours while incurring a much higher cost in electricity [21].

C. Monetary Payments

The final class of incentivizations offer rewards with what might be considered real money that holds external value. This approach is the one that moneTor adopts.

1) PAR: A pre-Bitcoin design, PAR [22] allow clients to send direct payments to relays in a hybrid payment scheme which makes use of inefficient but anonymous Chaumian e-cash protocols [23] and efficient but transparent probabilistic micropayments. PAR introduces the *honest but curious* bank paradigm in which the bank cannot deanonymize clients but is in control of their deposited financial assets. As with BRAIDS, PAR suffers from an unscalable centralized design.

2) ORPay, PlusPay, CoinPay: These three protocols, part of the Chaumian e-cash tradition of payments schemes, are a series of incremental improvements released across two papers [24], [25]. In each of the designs, the micropayment building block is derived from Payword hash chains [26], as is ours. While the schemes offer practical advantages to many aspects of PAR, they are limited by the same *honest but curious* security assumptions and their scalability remains capped by the need for all clients to interact with the central bank for each payment chain.

In the following presentation of moneTor, we will see how our approach resolves the scalability problem which all those schemes suffer by designing protocols based on payment channels.

IV. Economic Design

We first summarize the economic layer of the incentivization scheme. At this level, we assume the existence of an ideally secure and efficient payment layer and proceed to outline the high-level policy design. The challenge here is to engineer the flow of money in a way that is both economically stable yet still adheres to the core mission of the Tor Project.

A. Economic Policy

The moneTor scheme allows for relays to offer a *premium bandwidth* product to Tor users in exchange for monetary payments. Under this framework, financially willing users send payments directly to each relay along their circuits in exchange for higher internet bandwidth and faster download speeds relative to unpaid users. The moneTor tokens may be any form of programmatic money which satisfies the standard properties of *scarcity*, *fungibility*, *divisibility*, *durability*, and *transferability* [27, p.3] A schematic of the cash flow cycle is illustrated in Figure 1.

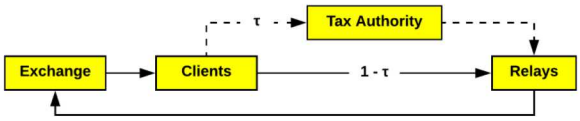


Fig. 1: Cash Flow — movement of moneTor tokens through the network. The value τ denotes the fraction of money that is collected for taxation purposes.

We introduce a novel concept within the field in the form of a taxation element. Intuitively, the shape of the network that will emerge in a purely profit-seeking environment may not perfectly correspond with the central goals of the Tor Project. The Tor Project may for instance desire to compensate certain types of relays more than others in order to improve the overall quality of the network (e.g., supporting areas where anonymity is particularly important but financial resources scarce). To this end, we accommodate for a stream of taxed income that is anonymously diverted into a shared pool of funds. These funds are selectively redistributed to relays via a transparent policy. In essence, taxation provides a tunable control mechanism for The Tor Project to shape the topology of the network towards some notion of optimal diversity and performance. The exact content of such policy is an active subject of research that is orthogonal to our paper (e.g., Waterfilling [28] argues for security by maximum diversity in endpoints of user paths, and TAPS [29] argues for security by trust policies).

A key economic question to address is the issue of price determination. While it would be tempting to enlist some market-based mechanisms to set premium bandwidth prices, any price differentiation between clients or relays inevitably leaks more information. This leakage becomes more severe with higher granularity payment options as adversaries begin to use price to link payment channels and circuits. We therefore impose the constraint that all users should pay a single uniform price for premium bandwidth at any time t. This price may be set through a centralized calculation by the authorities or a more dynamical consensus vote reached by the network.

The payment system we present in the next Section V is independent of the token creation for which multiple strategies are available. For example, the moneTor tokens could be viewed as wrappers for some external financial asset that is converted at an exchange service with an 1-to-1 mapping. In this scenario, moneTor tokens are created when the users buy them with other coins and destroyed when the users sell them for other coins. A second example would be to let the

Tor project creates and destroys moneTor tokens at will. In such scenario, the directory authorities handle the tokens as a resource management system, but do not trade them against fiat currencies or other coins. Given the transferability of the moneTor tokens, we would expect a side-market to appear for which the Tor project would not be responsible. These two approaches have both advantages and inconveniences usually linked to political opinions and ideologies.

B. Incentivized Conformity

In a decentralized network, there is no practical way to enforce standard behavior at each local node. We must therefore consider whether all nodes are rationally incentivized to obey the stipulated policies. For instance, we cannot guarantee that relays will actually confer premium bandwidth to paying users. Even though the relay has no particular reason to deviate, the client should periodically monitor her bandwidth and only make payments when they appear to be making a difference. The relationship between the client and relay can then be modelled as a game theoretic tit-for-tat dynamic.

At the opposite end of the spectrum, relays might overly prioritize premium circuits while rejecting all traffic from unpaid users. They might also attempt to game the tax redistribution process to gain larger share of the proceeds. The bandwidth measurement authorities must anticipate such modes of deviation from the standard behavior to ensure that the risk for a relay to get blacklisted from the network is greater than the incremental gains it might attain from cheating. These attacks, while manageable, suggest that it would be prudent to limit the complexity of our economic policies until we can better study behavioral deviation dynamics in the live network.

V. Payment Design

Where we previously assumed the existence of an ideal payment procedure, we now describe the concrete technical construction for the moneTor payment scheme.

A. Ledger

In our payment design, we follow the Bitcoin paradigm in which all users maintain full and exclusive control of their monetary wealth through use of public key cryptography [8]. However, unlike Bitcoin, it is unnecessary for moneTor to rely on an inefficient decentralized consensus mechanism. Since Tor already relies on a centralized set of authorities, we simply introduce a new ledger authority role to maintain the global payment state on a public tamper-evident database [30]. To maximize availability, this ledger may also be distributed across several authorities with, for example, a particular instance of RSCoin [31] compatible with our payment system.

B. Relay Payment Protocols

In this section we specify formal protocols that comprise the payment scheme. All protocols are either two or three party interactions between a subset of the following roles: C (client), R (relay), E (end user: either a client or relay), I (intermediary), and L (ledger). At this level, we assume that one client is paying one relay through a single channel and that communication lines will be appropriately anonymized by the Tor networking protocol.

Bolt The moneTor payment scheme is an extension of Bolt's three-party bidirectional micropayment channels. As such, we adopt its nomenclature where possible but adapt it to our concepts of ledgers, clients, and relays. The following is brief outline of the prerequisite micropayment channel procedures defined in Bolt. We refer the reader to the original paper for more detailed specifications [16].

- **KeyGen**: Any party generates a cryptographic keypair
- **Init-E**: E initializes half of a micropayment channel by escrowing funds on L
- **Init-I**: I initializes half of a micropayment channel by escrowing funds on L
- **Establish**: E and I interact to establish a new micropayment channel from their respective halves
- **Pay**: C interacts with I and R to send a single micropayment to R
- **Refund**: E closes a channel on L and makes a claim on the escrowed funds.
- **Refute**: I closes a channel on L and makes a claim on the escrowed funds.
- **Resolve**: L determines the final balance of funds awarded to each party.

While anonymous micropayment channels present a tremendous advance for many applications, the relatively heavy cryptography ($37 - 100\ ms$) and communication (7 message legs) is a prohibitive expense for Tor relay payments.[5] To overcome this barrier, we present a new payment layer design enabling far more efficient nanopayments that will satisfy our constraints.

moneTor The moneTor construction makes use of the existing anonoymous micropayment structure to facilitate *locally transparent nanopayments*. In this scheme, *nanopayment channels* are established in place of a single micropayment operation. These lightweight channels allow the client to send n number of unidirectional nanopayments to the relay. Each payment represents a fixed value δ, established at the start of the channel. The payments themselves are *locally transparent* in the sense that each nanopayment in the same channel is trivially linked to each other. However, the nanopayment channels themselves are unlinkable to other nanopayment channels and micropayment operations. It is by design that this channel anonymity model fits well with Tor's existing circuit framework in which messages within circuits are linkable with eachother but the circuits themselves are not. In this sense, the overarching motivation of our work is to relax the costly anonymity guarantees provided by Bolt toward the design of a new set of protocols that have been optimized for Tor. The new protocols are briefly described in this section.

Any two parties C and R can construct a nanopayment channel once both have completed $Establish$ with a common intermediary I. We define the following set of protocols needed to manage nanopayments.

[5] In addition to concerns regarding global network overhead, it is also desirable to keep the barrier of entry low for smaller relay operators.

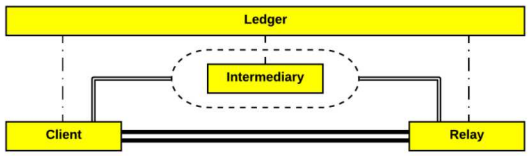


Fig. 2: Payment Roles — Dashed lines represent periodic transactions, thin double lines indicate micropayment channels, and thick double lines indicate a nanopayment channel. The dashed outline around the intermediary represents a notion of payment anonymity for the end users. Connections to the ledger and to the intermediary are protected by an internal Tor circuit.

- **Nano-Setup** C and I interact to prepare an incomplete half of a nanopayment channel on top of their existing micropayment channel.
- **Nano-Establish** C sends her nanopayment channel information to R, who interacts with I to complete the second half of the nanopayment channel on top of their own existing micropayment channel.
- **Nano-Pay** C sends a single nanopayment channel to R. This is repeatable for up to n operations.
- **Nano-Close-R** R closes his nanopayment channel with I.
- **Nano-Close-C** C closes her nanopayment channel with I. Note that this must happen after *Nano-Close-R*.

We also specify the following modified channel conflict resolution procedures to ensure secure closure properties for the nanopayment scheme.

- **Nano-Refund** E closes the channel on L.
- **Nano-Refute** I closes the channel on L.
- **Nano-Resolve** L makes final determination on both micropayment and outstanding nanopayment balances.

The core of our nanopayment scheme is inspired by the classic *Payword* two-party micropayment scheme in which payments are encoded by successively revealed preimages in a precomputed hash chain [26]. Hash chains are perhaps the most efficient known method for representing payments. In contrast to the expensive zero-knowledge proofs and signatures involved in anonymous micropayments, hash chain payments can be computed on the order of millions of hashes per second and conferred with a single 256 bit message, or one Tor cell.

The challenge in this construction is to securely integrate the hash chain concept into an existing three-party anonymous micropayment channel setup such that all parties maintain secure cryptographic ownership of their funds at all steps. At the same time, we must ensure that no deanonymizing information is leaked outside of the nanopayment channel context. We proceed to present a concrete scheme which incurs an overhead penalty of approximately two micropayment operations per nanopayment channel, one at the beginning and one at the end of the channel life cycle.

C. Nanopayment Protocols

In this section, we provide a summarized intuition for the basic steps in the payment protocol. A more formal treatment of the steps are provided in Appendix A with algorithmic details. Security considerations are detailed next in Section V-E and formally described in Appendix A.

Nano-Setup At the start of this protocol, C has access to a micropayment wallet w obtained from Bolt's **Establish** that enables her to operate her micropayment channel with I as well as a refund token rt that entitles her to claim her current funds on L should I misbehave or go offline. To construct a nanopayment channel, C first generates an array of values hc of length n where $hc_i = H(hc_{i+1})$ and hc_n is a random number. The root of the hash chain hc_0 is used to create a globally unique nanopayment token nT that encodes the public parameters of the channel including the length n and the per-payment value δ. C sends I a commitment to a fresh nanopayment channel parametrized by nT along with a zero-knowledge proof of the following statements:

1) The nanopayment wallet nw is well-formed from w
2) C has ownership of a micropayment channel containing at least $n * \delta$ funds.

I verifies these messages and supplies C with a new signed refund token nrt that entitles C to cash out the full balance of the micropayment channel. C, now protected against misbehavior by I, agrees to send a revocation token σ_w, which revokes her right to use w or rt. I is now protected against double spending by C and can safely inform C that the nanopayment channel has been set up successfully.

Nano-Establish At this point, C sends R the same nT token used to setup the channel with I. R uses the token to initiate her end of the nanopayment channel with I by executing essentially the same procedure that C used in *Nano-Setup*. The nanopayment channel is now fully established and ready to be used. A key observation is that both ends of the channel (C-I and R-I) are rooted at the same hash chain root hc_0.

Nano-Pay To make the next i^{th} payment, C simply sends the next hash preimage hc_i to R. Knowledge of this preimage hc_i is sufficient for R to prove possession of a nanopayment. At any given time, R can broadcast the tuple (nrt, hc_i) to L to prove ownership of the correct balance of funds. Notice that this action simultaneously reveals hc_i to I, who can then claim an equivalent value of funds from C. As a result, the scheme satisfies a correct-by-construction property of *atomicity* whereby both legs of the protocol are finalized at the same time.

Nano-Close After some number of payments $k < n$ has transpired and C wants to close the Tor circuit, both C and R will generally prefer to close their nanopayment channels through I. In this process, the R-I leg must be closed before the C-I leg. This is due to the unidirectional nature of nanopayment channels. Since payments are flowing form C to R, I must first determine its debt to R in order to know how much it can claim from C.

R first sends to I a commitment to a new micropayment wallet w' and a zero-knowledge proof of the following statements:

1) w' is well-formed from w (w was either created by Bolt's establish phase or by a previous moneTor Nano-Close)

2) The balance of w' is equal to the sum of the balance from the previous wallet w and $\delta * k$

Once verified, I issues a refund token rt' on the new funds. R agrees to invalidate the nanopayment channel by issuing a revocation token σ_{nw} to I. I and R proceed to create a blind signature on w' thus validating the wallet for future use.

Once I has closed his nanopayment channel leg with R, I and C are free to complete the exact same close protocol. All parties are now reverted to the original state they occupied prior to *Nano-Setup* save for a securely updated balance.

Nano-Refund, Nano-Refute, Nano-Resolve Honest parties will not typically close active nanopayment channels on the ledger, opting instead to run Bolt micropayment closure procedures when they wish to cash out. However, in the event of malicious behavior or premature abortion, *Nano-Refund* and *Nano-Refute* outlines procedures for E and I to withdraw funds on the ledger at any given step with the latest payment information. After a set amount of time allowing for the counterparty to reciprocate, the ledger runs *Nano-Resolve* to make a final publicly verifiable determination on the final balance. Correct execution of these procedures allow all honest parties to retain their funds in some cases and obtain the full balance of the malicious party's escrowed funds in others.

D. Tor Integration

Up to this point, we have described payments that occur between a single client and a single relay. In practice, it is typical for each client to maintain a handful of concurrently active circuits,[6] each of which requires three streams of payments to the guard, middle, and exit relays. These channels must be actively managed to optimize computational overhead as well as money flow. Furthermore, connections between the client and the guard relay are transparent and persistent across the timescale of several months. We optimize toward this setup by enabling transparent and direct payment channels between the client and guard, considerably reducing the network overhead costs.

The zero-knowledge setup also allows an elegant way to anonymously handle the tax collection policy described in Section IV. Thus far, we have treated the nanopayment value δ as symmetric for both the client and relay leg. In practice, it requires only a trivial modification to specify separate values of δ_C and δ_R such that the following equality is satisfied.

$$\delta_C - \delta_R = tax + fee \tag{1}$$

Here, tax is the portion of every payment that is redirected to the Tor tax authority while fee represents compensation for I's services. I gradually accumulates these overhead charges in his balance over the course of running many nanopayment channels. When it is time for I to cash out the full micropayment channel, L simply divides the funds between the I and the tax authority. Note that this does *not* mean that L can arbitrarily control money as this process is well-defined in setup of the network protocol.

[6]For instance, the popular Tor Browser user application typically does not share circuits with streams targeting a different destination address unless those streams come from the same SOCKS connection

E. Payment Security and Anonymity

Our security model must account for both privacy and payment security. The privacy threat model is derived from the local active adversary paradigm ubiquitously found in Tor research [1]. Like all cells in Tor, Nanopayment messages are locally linkable by relays participating in the circuit. However, since each circuit is only ever associated with one anonymous micropayment channel at any given time, we can guarantee that relays and intermediaries cannot link two nanopayment channels with the same user. Formal definitions, theorems, and proofs are provided in Appendix A.

Observing the cash-out transactions on the ledger does not reveal any information regarding the source of the payment. Moreover, any communication in our tripartite nanopayment protocol are protected by Tor circuits. Hence, relationship between client to intermediary, client to ledger, relay to intermediary and relay to ledger are themselves anonymized by Tor circuits. If some party aborts, which induce a dispute on the ledger, the relationship between the client and his relays is protected by the various Tor circuits built to resolve the dispute with the ledger.

Informally, we claim the following anonymity guarantees relative to unmodified Tor:

1) Additional parties needed to operate the moneTor system (i.e. ledgers and intermediaries) cannot extract any more information about a given client than any middle relay.
2) Excluding side channels, circuits do not leak any more information than the single bit needed to differentiate premium and nonpremium users.

The unmodified construction exposes a subtle passive attack by I. By examining the final number of payments made on each channel in conjunction with the globally fixed nanopayment cost, I may potentially link all of C's nanopayment channels.[7]. To mitigate this vulnerability, we stipulate that C must make a least one micropayment, which has a monetary value hidden from I, before closing a micropayment channel. We expect that this micropayment should contain a random value not greater than the channel escrow maximum value as stated in the Tor consensus and may be made to another account owned by C.

Our threat model for payment security is similar to those found in prior works in blockchain micropayment channels [14]. In such models, the user is protected from malicious intermediaries by the ability to prove misbehavior to a global ledger. The security of the ledger will be subject to the guarantees made by the cryptocurrency interface [32], [33]. This is in direct contrast to prior Chaumian e-cash proposals that employ the *honest but curious* model for the central bank in which banks have full control over user funds. Indeed, this is the basis for the difference in terminology between prior *central banks* and the moneTor *ledger*.

[7]This attack is best illustrated with a trivial example. Suppose that I facilitates a number of nanopayment channels with the following number of payments, each of which is known to represent one unit of money: $[58, 839, 356, 881, 23, 89, 561]$ Now that C closes her micropayment channel and terminates with exactly 404 units of money. Once the micropayment channel is closed, I must necessary gain knowledge of the final balance of funds and can easily link the first and third nanopayment channels as belonging to the same C

VI. Network Design

A. Extending the Tor protocol

We extend the Tor routing protocol described in Tor's specifications [34] and we exploit Tor's leaky-pipe circuit topology[8] to exchange payment information with each hop of the circuit. We introduce two new control cell types: one link-level cell and one relay-level cell. The link-level cell is used to exchange information related to the payment protocol between the Tor client and its guard relay while the relay-level cell is used for the middle relay and the exit relay. This subtype of relay cell is comprised of a payment header denoting the type of payment cell, followed by a payload of payment data. To an outside observer, payment cells are indistinguishable to normal relay cells. Figure 3 shows the internal structure of the cell.

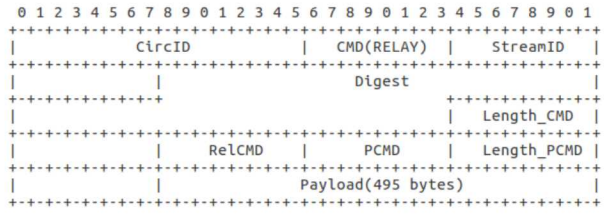


Fig. 3: Relay Payment Cell — Cell definition specifying the structure of a moneTor payment cell. Note: a block that appears blank and empty is in fact the continuity of the previous row

All the bytes starting from StreamID (included) are onion encrypted. RelCMD is set to RELAY_COMMAND_MT, PCMD is the payment command which is different for each step of the payment protocol. If some message overflows the payload available length (495 bytes), we queue multiple cells of the same PCMD and buffer them on the receiver side to unpack the whole message.

B. Pre-built Channels

By default, Tor attempts to pre-build circuits in order to reduce latency once a user wishes to create a data stream. Much like circuits, moneTor payment channels are high in initial latency because of the many in-out messages in the protocol. We therefore exploit the same strategy currently used in circuit establishment by allowing payment channels to be preemptively set up and established on clean pre-built circuits. This dramatically reduces the effective time to first payment. Unfortunately, excessive establishment of preemptive channels will eventually afflict the network with unused overhead. Our implementation features a rudimentary prediction strategy that attempts to balance this trade-off by anticipating the number of needed channels using historical usage information in a similar way Tor anticipates the need for a fresh circuit.

[8]"Leaky pipe" refers to the ability of the user to direct traffic that ends at an intermediate hop along the circuit

C. Network Scalability

In our design, we are concerned with memory consumption, kernel socket consumption and CPU consumption. Our choice for a tripartite protocol effectively shifts the memory consumption of opened and idle micro channels to the intermediary nodes of the network. A more basic setup whereby each Tor client maintains a micropayment channel with each relay would incur an $O(n*m)$ cost with respect to channel management complexity, with n the number of Tor clients and m the number of relays. By engineering an additional intermediary layer, the complexity of moneTor channel connections is reduced to $O(n+m)$. In our implementation, intermediary relays do not participate in routing user streams and are tasked only with providing payment channel services. Intermediaries devote the full extent of their computational resources toward this task, allowing only a few strong nodes to handle all of channel management needs of the network.

Interactions between parties are realized within Tor circuits to allow multiplexing of circuits over the same TCP connection. This also protects the identity of the client and its chosen circuit against identification by the ledger or the intermediary[9]. As a result, the intermediary and the ledger must have a number of available sockets higher than the number of relays in the network in the worst case. Since this limit is in line with Tor's current assumption, our design inherits the same socket consumption scalability of the greater Tor network.

D. Prioritized Traffic

The final component of our network-oriented design addresses the need to deliver prioritized bandwidth given an explicit signalling of premium or nonpremium traffic. Our objective is to provide a tunable range of prioritization while incurring as little cost as possible to average global performance. In our design, the chosen value is enforced network-wide by the directory authorities in order to avoid partitioning of the anonymity set. Traffic scheduling is perhaps the most intuitive mechanism with which to implement prioritization. However, we found in our investigation that practical modification of the Tor scheduling infrastructure is nontrivial. A more detailed discussion of our findings can be found in Appendix A.

Fortunately, Tor's overlay flow control mechanism provides an alternative route to implement our desired functionality. Recall that edge nodes regulate the traffic flux in either direction using a set of flow control windows. Roughly speaking, these windows determine space allotted to each circuit on a relay's scheduling queue, which in turn is positively correlated with effective bandwidth. We implement our prioritization scheme by readjusting the windows according to the following formula.

$$window' = window(1+\alpha(premium/premium\%-1)) \quad (2)$$

Here, a circuit is marked as prioritized by the bit $premium \in \{0,1\}$. The tunable priority benefit $\alpha \in [0,1)$ defines the proportion of the non-premium capacity that we wish to

[9]We assumed no side-channel exploitation in this work but do discuss timing attacks in Section IX

transfer to premium clients. By accounting for $premium\%s \in [0, 1]$, the fraction of premium to nonpremium clients, we can keep the total flow capacity constant. Our policy may be implemented in one of two ways. First, each node could track the $premium\%$ locally and dynamically adjust their own windows. This introduces a considerable amount of added complexity with unclear consequences on network performance. A more sound approach calls for the Tor authorities to track the global value for $premium\%$ and periodically broadcast static flow control windows to be used by the entire network. We adopt the latter approach in this iteration of moneTor.

Interlude.: This concludes the design of the moneTor framework. In the proceeding sections, we describe steps taken to iteratively select and validate key parameters as well as the scheme as a whole. Such parameters include: payment frequency, preemptive channel creation, and prioritization amounts (α). Throughout this process, the underlying objective is to prove that we can confer qualitatively "significant" advantage to paid premium users while incurring minimal overhead costs with respect to throughput, memory usage, and latency within a realistic network environment.

VII. Empirical Analysis

A. Data Collection

We deployed during two weeks a data collection system to look for empirical temporal information about lifetime and bandwidth consumption in Tor circuits. Our objective was to have a deeper understanding of typical Tor usage and whether such usage can benefit from our channel-based payment system. For example, these measurements might capture some notion about the type and magnitude of potential premium traffic. We classify the traffic type based on the service connection port. Besides the classical ports 80 and 443 used for web traffic, we aggregate data from some other families including the WHOIS protocol [35] and RWHOIS [36] from ports 43 and 4321, respectively. The complete list of families is constructed from the reduced exit policies which we run on our relays. This measurement methodology allows us to reason based on application specific traffic.

1) Efforts to preserve users privacy: To ensure ethical experimentation, we first contacted the Tor research safety board [37]. The feedback we received was subsequently used to refactor our data collection process (e.g., no metadata from any specific flow or circuit should be written on disk).

Data from five old and very stable exit relays with a total bandwidth of 50 MiB/s was collected, stripped of origin metadata, and aggregated on a central server. The data collected from each relay is itself an aggregation which we perform inside the relay's memory. The aggregation is done only for "active circuits" from the exit relay perspective. We consider a circuit active when it has received a connection request to an IP address on the internet. The aggregation is done inside bins of configurable size for every different traffic family we consider. Once we collected enough data from a single family, we dump the information on the disk, clear the data, and resume a new session. The final information obtained contains

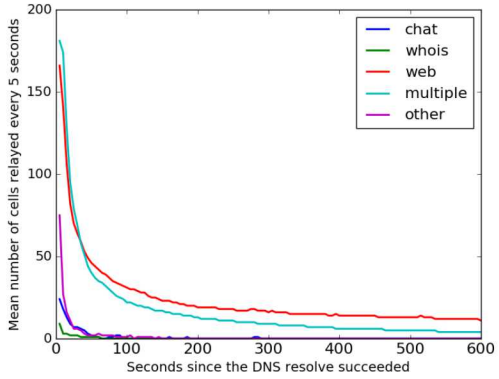


Fig. 4: Time Profile — Average distribution of traffic across circuit lifetime beginning with the first DNS request.

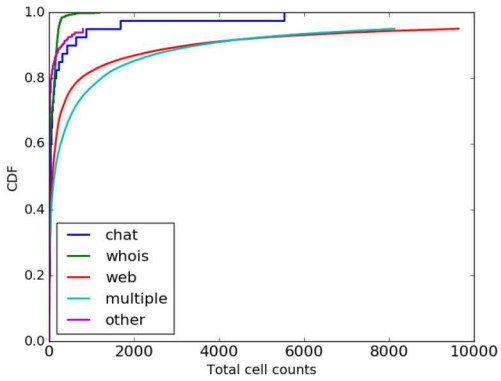


Fig. 5: Total Counts — Distribution of circuit size with respect to the total number of cells processed

an aggregation of 1600 circuits over an unspecified time frame that is implicitly determined by the rate of user activity. The following data was considered:

- **Time Profile**: The number of cells in each time interval (configured to be 5 seconds) since the success of the DNS request. This information sums inbound and outbound cells, and is aggregated over circuits by addition.
- **Total Counts**: The total amount of cells processed by a circuit. This information is aggregated by taking the mean of fixed-size nearest neighbor bins.

Crucially, we do not record information linked to any single particular user flow on disk.

2) Observations: Our measurements successfully captured several important pieces of information for the design and justification of moneTor. For example, one important task is to determine the number of potential users that could benefit from paid traffic. From Figure 5, we observe that $\approx 82\%$ of circuits carrying only web traffic exchanged less than 1000 cells. While we cannot deduce any statements about users, we can speak to the fraction of circuits that may benefit from a payment channel in the Tor network, since around 50% of them do not carry data and less than 17% of them carry at least one web page. The remaining 18% would appear to be better candidates for moneTor.

It is also evident from Figure 4 that most of the traffic is usually carried within the first few tens of seconds, and that all types of traffic we collected seems to follow the same rule. From that result, we believe that the reliability of payment is critical within the first few seconds, especially from a relay viewpoint. Our payment channels should ideally be established and ready before the user begins to use the circuit. While this result cannot be guaranteed for an unbounded number of circuits, a well-designed preemptive circuit build strategy should do a sufficient job of eliminating channel setup/establish latency in the average case.

VIII. Simulated Validation

Having established the empirical context for a channel payment scheme, we validated our technical design via experiments performed on a prototype software implementation within the native Tor codebase.

A. Prototype

A substantial contribution of the research is embedded within our implementation of the moneTor framework. The modifications, applied to Tor release version 0.3.2.10, cover approximately fifteen thousand added lines of code across Tor's core C software. We emphasize that the implementation is optimized solely for our experiments. Most notably, we simulated cryptographic operations such as zpk proofs and commitments in the nanopayment creation and close procedures using CPU delays. Furthermore, those delays are implemented in account for Shadow CPU delay design: if the Tor process is considered to run over a multicore computer, the costly operations are simulated over a dedicated core. If the Tor process is considered to run over a singlecore computer, then the costly operations are simulated by an appropriate number of dummy AES encryptions in a thread concurrently to the Tor process main thread. Those AES encryptions are captured by Shadow and replaced by CPU delays on the virtual CPU, affecting Tor's main thread. These delays were tuned to conservatively reflect real measurements in background work [16].[10] Note that our prototype does not implement anything that does not help us to answer our research goals, such as coin/wallet management, extension of the Tor control protocol to manage the asset and various options, Intermediary information recovery in case of crash, etc. The prototype serves the following purposes in our study.

1) An implementation covering nuances not explicitly covered in the protocol designs. In effect, we would like to show that there are no unexpected and prohibitive practical conflicts with the existing Tor design.
2) A platform to study the feasibility of premium circuit prioritization from a networking perspective.
3) A platform to obtain a rough factor-of-two approximation for all bandwidth, computation, and memory requirements of the system, both globally and at individual nodes.

[10]Extracted values are conservative in the sense that our zero-knowledge proofs require proving only a subset of the statements required in each corresponding Bolt zero-knowledge proof.

The first design purpose is clearly qualitative and we briefly note that we did not discover any insurmountable logical flaws in the design. To analyze the networking dynamics and resource consumption, we studied our implementations through the following proceeding experiments.

B. Methodology

Experiments were conducted using the Tor shadow simulator tool [10]. We ran two sets of experiments at different scales from a consensus document published in early February 2018. The first set featured 100 relays, 1000 clients, 10 intermediaries, and ran for a total of 90 minutes. These experiments were used to gather information concerning the system overhead and protocol execution times. The second set featured 250 relays, 2500 clients, 25 intermediaries, 80 minutes of total run time, and was used to measure the performance benefits conferred to premium clients. In both cases, simulated traffic features 16% *bulk* clients who continuously download 5 MiB files and 84% *web* clients who periodically download 2 MiB files,[11]. The number and behavior of clients were chosen to satisfy (A) realistic congestion rates measured by a transfer timeout percentage around 4% [2] and a historical bulk/web global traffic ratio of about 50% [39], [40]. Importantly, it should be noted that neither the scale of our experiments nor the precise configuration of client nodes are intended to be precise replicas of real-world conditions. Tor networking is itself a complex area of research and we are content to adopt the simplest model that will highlight the relatively crude networking needs of our incentivization scheme.

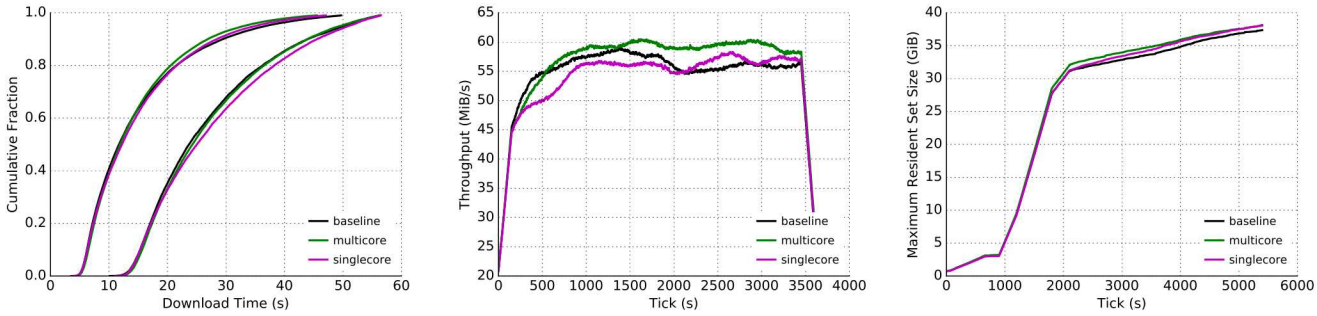
C. Experiments

Our experiments are separated into three groups each capturing a separate characteristic of the scheme.

1) Global Overhead: First, we attempt to show the total cost the moneTor scheme in terms of total network throughput. To highlight worst-case performance, we configured a medium scale experiment consisting of 100% of premium clients and compare to a baseline trial with 0% premium clients. No actual traffic priority is conferred. Since our algorithms makes use of some concurrent crytpographic operations, we are concerned with the number of CPU cores available to most relays. This information is not publicly available. As a result, we provide two trials with moneTor: one where we assume all nodes are running on multi-core hardware and one in which all nodes are running on single-core hardware. The results are summarized in Figure 6.

Our findings indicate that even in the worst case scenario, our system incurs statistically negligible overhead at these scales across all three measures of download time, throughput, and memory usage. This in line with the raw bandwidth overhead observations in which a small fraction ($< 1\%$) of the network traffic is attributable to moneTor payment messages. This is true for all of our trials which feature a payment rate of 1 payment (1 cell) every 1000 data cells exchanged in either

[11]While 5 MiB bulk files are a common standard in Tor benchmarking [2], 2 MiB web files reflect the approximate size of modern web pages [38]



(a) Download Time Overhead (b) Throughput Overhead (c) Simulation Memory

Fig. 6: Global Overhead — Comparison of overhead in pure multicore and singlecore network conditions. Figure 6a shows two sets of download time CDF curves for each file size (2 MiB and 5 MiB), Figure 6b shoes the 5 minute moving average throughput over time, and Figure 6c shows the memory consumption across the experiment lifetime. The simulation includes 100 relays, 2 authorities, 1 ledger authority, 10 intermediaries and 1000 Tor clients scaled down from the public consensus file '2018-02-03-00-00-00-consensus'.

direction. It is also possible to increase the payment rate for more fairness if needed, as long as the total overhead induced from control cells is kept under an acceptable fraction of the overall data bandwidth.

2) Payment Latency: Given results from our data collection, we surmise that payment latency is a crucial factor in servicing our front-loaded clients. To this end, we measure the distribution of completion times for various steps in the protocol. The results shown here are collected from a worst case multicore experiment featuring 100 relays and 1000 clients all of whom run premium circuits. To highlight the effects of native latency in the Tor network, we show payments split across each relay role of guard, middle, and exit. Recall that moneTor makes use of high-overhead, low-marginal cost payment channels. The bulk of the cost in our scheme lies in the conduction of the nanopayment channel *establish* and *close* protocols as shown in Figure 7a and Figure 7c. Notice that close operations are about twice as time consuming as establish operations reflecting the need for the relay to close his half of the nanopayment channel before the client can complete hers. Figure 7b illustrates time to first payment, our most revealing latency metric. This measure includes the overhead in channel establishment when we do not have available preemptive channels. In the best case scenario, when all three payment channels have been correctly pre-built for the circuit, this measure is equivalent to a single trip toward each relay. Comparing this Figure 7b to Figure 7a, we observe the effectiveness of preemptive channel building.

In all protocol phases, we observe that latency for guard relays are negligible in comparison to the middle and exit relays, further validating our design decision to implement special directly paid guard channels.

3) Network Priority: Our final set of experiments studies the success of our scheme in delivering prioritized traffic for premium users. To perform this analysis, we prepared sets of three small experiments with varying modifier priorities $\alpha \in \{0, 0.25, 0.5\}$, where $\alpha = 0$ represents the baseline control. Our first set of experiments assuming 50% of premium users is shown in Figures 8a, 8b, and 8c. From this illustration, is it evident that premium users enjoy an advantage in internet speed compared to nonpremium users and that this advantage is more pronounced with the higher priority modifier. Moreover, the evenly split premium and nonpremium performance "averages out" to approximately mirror the baseline experiment, indicating little loss in overall network performance, and confirming our overhead experiment.

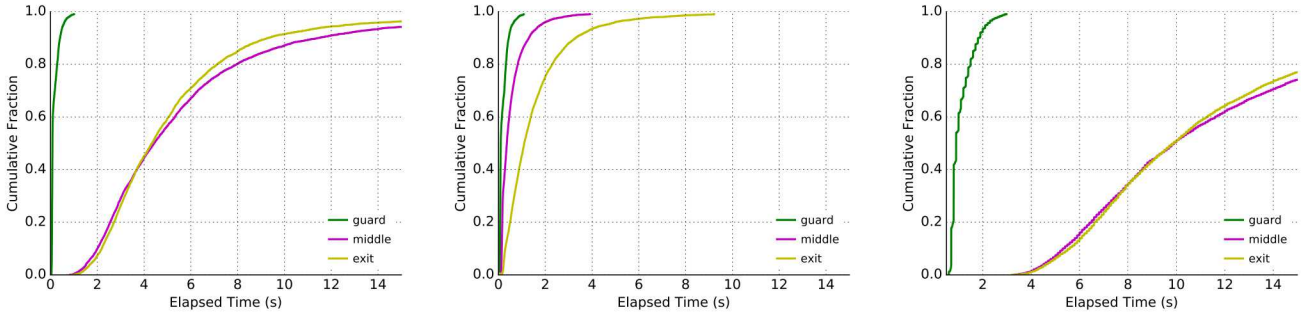
Note, one difference of our priority mechanism over a scheduling approach is that the latency is unchanged: there is no reason for the first byte to come faster for premium users since our priority mechanism only change the size of the control flow window (i.e., how many cells the circuit is allowed to carry at any time, see again Section VI-D for details). Such approach would keep the network responsive for nonpremium and low-bandwidth usage, while still offering an upward benefit for premium users while loading web pages, as shown in Figure 8.

Our first analysis on Figures 8a, 8b, and 8c assumes 50% of premium users. Reducing that fraction to 25%, which we assume to be a more realistic model should most likely, according to Equation 2, expect an even greater benefit on a per-user basis. Figures 8d, 8e, and 8f shows that this is indeed the result. Users across the spectrum of web speed enjoyed $\approx 100\%$ improvements in download speeds relative to nonpremium users — likely enough to induce some amount of monetary exchange. Note that the parameters from Equation 2 can be tuned (enforced network-wide) to offer more or less benefit for premium users on the willingness of the directory authorities.

IX. Limitations and Future Work

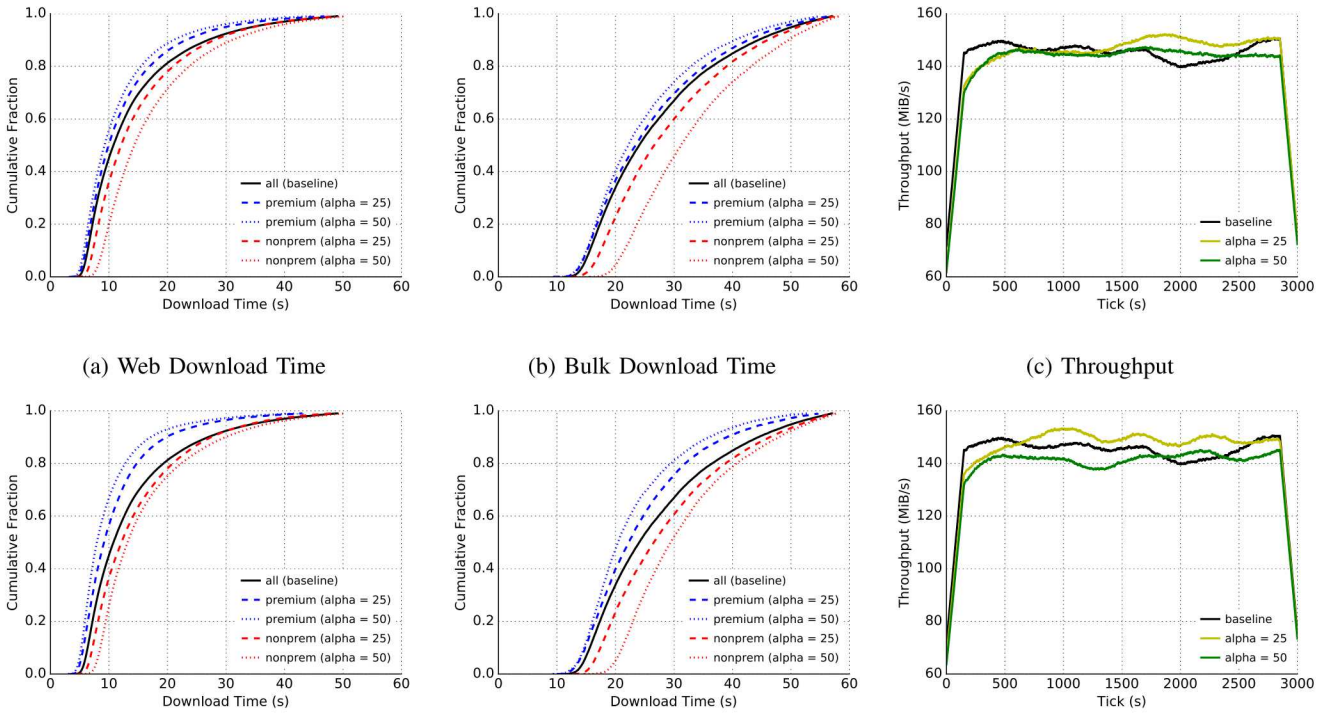
A. Limitations

The moneTor framework faces a number of technical and economic limitations. Firstly, the mere presence of paid traffic leaks at least a single bit of information to relays and intermediaries regarding the browsing habit of the user. In



(a) Nano-Establish (b) First Payment (c) Nano-Close

Fig. 7: Protocol Execution Time — Time to finish each protocol step split across interactions with each of the three relay. The simulation includes 100 relays, 2 authorities, 1 ledger authority, 10 intermediaries and 1000 Tor clients scaled down from the public consensus file '2018-02-03-00-00-00-consensus'.



(a) Web Download Time (b) Bulk Download Time (c) Throughput

(d) Web Download Time (e) Bulk Download Time (f) Throughput

Fig. 8: Prioritization Benefit — Performance differentiation between paid and unpaid users. The first row displays results with 50% premium users while the second row displays results for 25% premium users. Simulations feature 250 relays, 2 authorities, 1 ledger authority, 25 intermediaries and 2500 Tor clients scaled down from the public consensus file '2018-02-03-00-00-00-consensus'.

practice, we expect that this information might be positively correlated with certain user applications such as bittorrent or video streaming. Secondly, our nanopayment protocols leak the number and value of payments made to the intermediary. While this information cannot by itself be used to identify clients or relays, we expect that it could provide useful contextual knowledge when combined with other inputs. Finally, although we ensure payment privacy within the Tor ecosystem, we can make no such guarantees at the token exchange interface. Users who practice inadequate operational security risk leaking information about the net value of payments they make in Tor with their real-world identities. This problem may be of greater concern for users residing in political regimes that are politically opposed to Tor.

A separate financial limitation arises from our our tripartite payment channel architecture and the need for intermediaries to escrow large amounts of funds. This opportunity cost of the frozen capital in terms of forgone investments will be passed on as financial overhead to paying users in the form of intermediary fees.

B. Future Work

1) Prioritized Traffic: Although we were able to successfully demonstrate clear prioritized premium traffic using flow control windows, there is considerable room to analyze prioritization under more realistic settings. Among the possibilities, we maintain that scheduling provides the most conceptually elegant way to achieve our goals. In the absence of effective scheduling, extension and further analysis of our flow-control technique provides an alternative path forward. This future work might study methods for ensuring a more robust response to local variations in the fraction of premium users as well total network load.

2) Side-channel Attacks: We assumed that our protocol construction would not give more information to an observer than the single bit needed to differentiate a premium user. However in practice, the protocol may leak information through side-channels, such as the time needed to exchange cells in order to open/close channels. An intermediary, or an exit relay may reason about the client's location by evaluating the round-trip-times of the multi-rounds protocol. Although often easily mitigated once discovered, these side channels pose an interesting problem to our expanded attack surface.

3) Tax Redistribution and Adversary Reactions: The understanding of the right policies for tax redistribution is an ongoing research question and was out of scope of this paper. Further work might answer what type of behavior must be incentivized to comply with a particular goal, and how to prevent adversaries to game the system.

4) Channel Management: Given the dynamic activity of payment channels that we have shown, further research might explore more sophisticated policies to manage moneTor payment channels. Such policies would make decisions concerning micropayment channel initial value, nanopayment channel size, preemptive channel creation, and the selection of available channels to apply to each new circuit. An ideal policy would be responsive to network load and predicted user behavior. In addition to the classical optimization objectives of computation, memory, and bandwidth, this research effectively considers a brand new vector as it attempts to streamline the flow of money through the network.

5) Extended Applications: Existing privacy-centric currencies are limited in that they are only effective when used in conjunction with an anonymizing network such as Tor [41]. Our currency scheme more explicitly integrates the two layers and. This, combined with the featured efficiency and scalability in particular makes moneTor an attractive option for such applications as metered video viewing, metered video gaming, and pay-per-page websites. The study and implementation of these applications in an anonymous setting provides an interesting opportunity for multidisciplinary research.

X. Data Reproducibility

Our code developed during this research and needed to reproduce our graphical results can be found in various repositories at @Monetor Github account [42].

XI. Conclusion

The Tor network suffers from concerns in both performance and diversity. In accordance with the general law that large networks are better than small networks, we assert that the quality of Tor along both of these vectors can be simultaneously improved by incentivizing more relay participation, while promoting some diversity notion. To this end, we present moneTor, a comprehensive framework for incentivizing Tor relay participation through true monetary payments. Our design broadly covers every major level of implementation from economic policies to the protocol payment layer and integration into the Tor networking architecture. Along the way, we developed novel protocols for highly efficient and cryptographically secure nanopayment channels as well as novel techniques in networking integration. A small venture into empirical data collection reinforced our intuition that existing user behavior is compatible with a light-weight payment incentive scheme. This led to our more involved round of experimentation in which we tested our natively integrated moneTor prototype. The results of this investigation were highly encouraging, indicating low latencies, negligible throughput overhead, and upwards of 100% benefits for paying users in the simulated environment.

Legal, political, and sociological questions concerning the prudency of introducing money into the Tor network are difficult to answer in a laboratory setting. However, this work indicates that our solution to Tor incentivizaiton offers promising properties on all measured technical and economic features. Pending further refinement, we optimistcally conclude that moneTor is the first scalable monetary payment framework for Tor that can be feasibly developed today.

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Naval Research Lab Washington DC, Tech. Rep., 2004.

[2] T. M. Portal, "Tor performance metr," 2018.

[3] M. K. Wright, M. Adler, B. N. Levine, and C. Shields, "The predecessor attack: An analysis of a threat to anonymous communications systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 7, no. 4, pp. 489–522, 2004.

[4] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 183–195.

[5] M. AlSabah and I. Goldberg, "Performance and security improvements for tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, p. 32, 2016.

[6] J. Reardon and I. Goldberg, "Improving tor using a tcp-over-dtls tunnel," in *Proceedings of the 18th conference on USENIX security symposium*. USENIX Association, 2009, pp. 119–134.

[7] R. Jansen, J. Geddes, C. Wacek, M. Sherr, and P. F. Syverson, "Never been kist: Tor's congestion management blossoms with kernel-informed socket transport." in *USENIX Security Symposium*, 2014, pp. 127–142.

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

[9] F. Rochet and O. Pereira, "Dropping on the edge: Flexibility and traffic confirmation in onion routing protocols," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 2, pp. 27–46, 2018.

[10] R. Jansen and N. Hooper, "Shadow: Running tor in a box for accurate and efficient experimentation," MINNESOTA UNIV MINNEAPOLIS DEPT OF COMPUTER SCIENCE AND ENGINEERING, Tech. Rep., 2011.

[11] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 3, pp. 3–12, 2003.

[12] M. AlSabah, K. Bauer, I. Goldberg, D. Grunwald, D. McCoy, S. Savage, and G. Voelker, "Defenestrator: Throwing out windows in tor," in *Proceedings of the 11th Privacy Enhancing Technologies Symposium (PETS 2011)*, July 2011.

[13] B. I. Team, "Blockchain info," https://blockchain.info/stats, 2018, accessed: April 2018.

[14] J. Poon and T. Dryja, "The bitcoin lightning network: Scalable off-chain instant payments," *draft version 0.5*, vol. 9, p. 14, 2016.

[15] Y. Zhang, Y. Long, Z. Liu, Z. Liu, and D. Gu, "Z-channel: Scalable and efficient scheme in zerocash," IACR Cryptology ePrint Archive 2017 (2017), 684. http://eprint. iacr. org/2017/684, Tech. Rep., 2017.

[16] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2017, pp. 473–489.

[17] R. Dingledine, D. S. Wallach *et al.*, "Building incentives into tor," in *International Conference on Financial Cryptography and Data Security*. Springer, 2010, pp. 238–256.

[18] R. Jansen, N. Hopper, and Y. Kim, "Recruiting new tor relays with braids," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 319–328.

[19] R. Jansen, A. Johnson, and P. Syverson, "Lira: Lightweight incentivized routing for anonymity," NAVAL RESEARCH LAB WASHINGTON DC, Tech. Rep., 2013.

[20] M. Ghosh, M. Richardson, B. Ford, and R. Jansen, "A torpath to torcoin: proof-of-bandwidth altcoins for compensating relays," NAVAL RESEARCH LAB WASHINGTON DC, Tech. Rep., 2014.

[21] A. Biryukov and I. Pustogarov, "Proof-of-work as anonymous micropayment: Rewarding a tor relay," in *International Conference on Financial Cryptography and Data Security*. Springer, 2015, pp. 445–455.

[22] E. Androulaki, M. Raykova, S. Srivatsan, A. Stavrou, and S. M. Bellovin, "Par: Payment for anonymous routing," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2008, pp. 219–236.

[23] D. Chaum, A. Fiat, and M. Naor, "Untraceable electronic cash," in *Conference on the Theory and Application of Cryptography*. Springer, 1988, pp. 319–327.

[24] Y. Chen, R. Sion, and B. Carbunar, "Xpay: Practical anonymous payments for tor routing and other networked services," in *Proceedings of the 8th ACM workshop on Privacy in the electronic society*. ACM, 2009, pp. 41–50.

[25] B. Carbunar, Y. Chen, and R. Sion, "Tipping pennies? privately practical anonymous micropayments," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 5, pp. 1628–1637, 2012.

[26] R. L. Rivest and A. Shamir, "Payword and micromint: Two simple micropayment schemes," in *International workshop on security protocols*. Springer, 1996, pp. 69–87.

[27] T. Crump *et al.*, *The Phenomenon of Money (Routledge Revivals)*. Routledge, 2011.

[28] F. Rochet and O. Pereira, "Waterfilling: Balancing the tor network with maximum diversity," *Proceedings on Privacy Enhancing Technologies*, vol. 2017, no. 2, April 2017.

[29] A. Johnson, R. Jansen, A. D. Jaggard, J. Feigenbaum, and P. Syverson, "Avoiding the man on the wire: Improving tor's security with trust-aware path selection," in *Proceedings of the Network and Distributed Security Symposium - NDSS '17*. Internet Society, February 2017.

[30] S. A. Crosby and D. S. Wallach, "Efficient data structures for tamper-evident logging." in *USENIX Security Symposium*, 2009, pp. 317–334.

[31] G. Danezis and S. Meiklejohn, "Centrally banked cryptocurrencies," *NDSS Symposium*, 2016.

[32] A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille, "Enabling blockchain innovations with pegged sidechains," *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 2014.

[33] J. Poon and V. Buterin, "Plasma: Scalable autonomous smart contracts," *White paper*, 2017.

[34] R. Dingledine and N. Mathewson, "Tor protocol specification," 2018.

[35] L. Daigle, "Whois protocol specification," 2004.

[36] S. Williamson and M. Kosters, "Referral whois protocol (rwhois)," 1994.

[37] "Tor research safety board," https://research.torproject.org/safetyboard.html, 2018, members: https://research.torproject.org/safetyboard.html#who.

[38] H. A. Team, "Http archive," https://httparchive.org/, 2018.

[39] A. Chaabane, P. Manils, and M. A. Kaafar, "Digging into anonymous traffic: A deep analysis of the tor anonymizing network," in *Network and System Security (NSS), 2010 4th International Conference on*. IEEE, 2010, pp. 167–174.

[40] D. McCoy, K. Bauer, D. Grunwald, T. Kohno, and D. Sicker, "Shining light in dark places: Understanding the tor network," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2008, pp. 63–76.

[41] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *Security and Privacy (SP), 2014 IEEE Symposium on*. IEEE, 2014, pp. 459–474.

[42] "Account holding monetor code," https://github.com/monetor, 2018.

[43] M. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," https://eprint.iacr.org/2016/701, 2016.

[44] C. Tang and I. Goldberg, "An improved algorithm for tor circuit scheduling," in *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 2010, pp. 329–339.

[45] R. Jansen and M. Traudt, "Tor's been kist: A case study of transitioning tor research to practice," *arXiv preprint arXiv:1709.01044*, 2017.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energys National Nuclear Security Administration under contract DE-NA0003525.

APPENDIX
ALGORITHMS

This appendix more fully describes the algorithms we design to operate moneTor nanopayment channels.

A. Conventions

We adopt the following conventions in our algorithms.

- All variable names in this section except those in $CreateWallet$ are globally unique.
- Variable subscripts denote a party or role ((I)ntermediary, (C)lient, (R)elay, (E)nd user).
- New nanopayment variables are prefixed with the character (n). All other variables reference a value from the original Bolt scheme, although the name might be altered somewhat.

- Payment values (ϵ, δ) are signed integers with respect to the end user. For example, δ_C is negative and δ_R is positive in the typical case where a client is paying a relay.

B. Variable Index

The follow itemizes variables used in the algorithm design. The first level of variables are used for actual cryptographic and accounting operations. These are bundled into groups of higher level variable names meant to represent abstraction concepts such as payment channels and party states. Only these high level variables are stored between protocol executions.

$nT = (\delta_C, \delta_R, n, hc^0)$ — Nanopayment Channel Token — Stores static, public information that defines a nanopayment channel including the payment values on both legs, the max number of payments, and the hashchain head. This can be passed around freely by all parties.

$ncsk_C = (nwpk_C, nwsk_C, HC)$ — Client Nanopayment Secrets — Includes a Public/private key pair which allows the client to setup and close a nanopayment channel and a precomputed hash chain to make incremental nanopayments

$nS_C = (k, hc^k)$ — Client Nanopayment State — Mutable state of the nanopayment; includes the count of payments made so far and the latest sent hash pre-image

nrt_C — Client Nanopayment Refund — Allows the client to make a claim to the ledger on escrowed money at any time. This refund is signed by the intermediary and conditioned on revealing the latest hash pre-image that the client claims to have sent.

nrc_C — Client Channel Closure Message — Final message that is posted to the ledger by the client to claim all funds of the micropayment channel including any completed nanopayments.

$nS_I = \{nT : channel\_state\}$ — Intermediary Nanopayment State — Map of all past and present nanopayment channels and the corresponding channel state. Possible states are:

S_I — State of the Intermediary. Used to avoid multiple creations of nanopayment channels over the same micropayment wallet in parallel.

- $\perp$ — failed attempt at setting up a nanopayment channel
- $setup$ — channel has been set up by C
- $established$ — channel has been established with R
- $closed||hc^k$ — channel has been closed and no further payments are allowed

$ncsk_R(nwpk_C, nwsk_C, \perp)$ — Relay Nanopayment Secrets — Includes a public/private key pair allows the relay to setup and close a nanopayment channel. Since relays cannot make payments in this setup, the last field is left blank.

$nS_R = (k, hc^k)$ — Relay Nanopayment State — See nS_C

nrt_R — Relay Nanopayment Refund — See nrt_C

nrc_C — Relay Channel Closure Message — See nrc_C

Algorithm 1 Create Wallet Helper function for creating a new wallet

function WAL($pp, pk_{payee}, w, \epsilon$)
  parse w as $(B, wpk, wsk, r, \sigma^w)$
  $(wsk', wpk') \leftarrow$KeyGen(pp)
  $r' \leftarrow$Random()
  $wCom' \leftarrow$Commit($wpk', B + \epsilon, r'$)
  $\pi \leftarrow PK\{(wpk', B, r', \sigma^w)$:
    $wCom' =$ Commit($wpk', B + \epsilon, r'$) $\wedge$
    Verify($pk_{payee}, (wpk, B), \sigma^w$) $= 1 \wedge$
    $B + \epsilon \geq 0\}$
  return $(wsk', wpk', wCom', \pi)\}$

Algorithm 2 Nano-Setup Protocol between a client and intermediary to create a new nanopayment channel from an existing micropayment wallet. Run prior to circuit setup.

procedure CLIENT($pp, pk_I, w_C, \delta_C, n$)
  parse w_C as $(B_C, wpk_C, wsk_C, r_C, \sigma_C^w)$
  if $B_C + (\delta_C * n) < 0$ **then**
    Abort() ▷ consider new micropayment channel
  $\epsilon_C \leftarrow \delta_C * n$
  $(nwpk_C, nwsk_C, nwCom_C, n\pi_C) \leftarrow$ Wal($pp, pk_I, w_C, \epsilon_C$)
  $\delta_R \leftarrow -(\delta_C + tax)$ ▷ the tax is a net profit for I
  $HC \leftarrow$MakeHC(Random(), n)
  $nT \leftarrow (\delta_C, \delta_R, n, HC[0])$
  Intermediary.Send($wpk_C, nwpk_C, nwCom_C, n\pi_C, nT$)
procedure INTERMEDIARY(pp, sk_I, S_I, nS_I)
  $(wpk_C, nwpk_C, nwCom_C, n\pi_C, nT) \leftarrow$Client.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $wpk_C \in S_I \vee nwpk_c \in S_I \vee \neg$Verify($n\pi_C$) **then**
    Abort() ▷ invalid wallets
  if $-\delta_C \neq price \vee \delta_R + \delta_C + tax \neq 0$ **then**
    Abort() ▷ incorrect payment prices
  $S_I \leftarrow S_I \cup \{wpk_C : \perp, nwpk_C : \perp\}$
  $nS_I \leftarrow nS_I \cup \{nT : \perp\}$
  $nrt_C \leftarrow$Sign($sk_I, refund||nT||nwCom_C$)
  Client.Send(nrt_C)
procedure CLIENT
  $nrt_C \leftarrow$Intermediary.Receive()
  if $\neg$Verify($pk_I, refund||nT||nwCom_C, nrt_C$) $= 1$ **then**
    Abort() ▷ invalid refund token
  $nS_C \leftarrow (0, HC[0])$
  $ncsk_C \leftarrow (nwpk_C, nwsk_C, HC)$
  $\sigma_C^{rev(w)} \leftarrow$Sign($wsk_C, revoke||wpk_C$)
  Intermediary.Send($\sigma_C^{recv(w)}$)
procedure INTERMEDIARY
  $\sigma_C^{recv(w)} \leftarrow$Client.Receive()
  if $\neg$Verify($wpk, revoke||wpk_C, \sigma_C^{recv(w)}$) $= 1$ **then**
    Abort() ▷ invalid revocation token
  $S_I[wpk_C] \leftarrow \sigma_C^{recv(w)}$
  $nS_I[nT] \leftarrow setup$
  Client.Send($established$)

Algorithm 3 Nano-Establish Protocol between a client, intermediary, and relay to establish the nanopayment channel between the client and relay. Run at the start of circuit setup.

procedure CLIENT(nT)
  Relay.Send(nT)
procedure RELAY($pp, pk_I, B_{I:B}, w_R$)
  $nT \leftarrow$Client.Receive()
  parse w_R as $(B_R, wpk_R, wsk_R, r_R, \sigma_R^w)$
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $B_{I:B} - (\delta_B * n) < 0$ **then**
    Abort() ▷ consider new micropayment channel
  $\epsilon_R \leftarrow \delta_R * n$
  $(nwpk_R, nwsk_R, nwCom_R, n\pi_R) \leftarrow$Wal($pp, pk_I, w_R, \epsilon_R$)
  Intermediary.Send($wpk_R, nwpk_R, nwCom_R, n\pi_R, nT$)
procedure INTERMEDIARY(pp, sk_I, S_I, nS_I)
  $(wpk_R, nwpk_R, nwCom_R, n\pi_R, nT) \leftarrow$Relay.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $wpk_R \in S_I \vee nwpk_R \in S_I \vee \neg$Verify($n\pi_R$) **then**
    Abort() ▷ invalid wallets
  if $nS_I[nT] \neq setup$ **then**
    Abort (unregistered nanopayment channel)
  $S_I \leftarrow S_I \cup \{nwpk_R, \perp\}$
  $nS_I[nT] \leftarrow established$
  $nrt_R \leftarrow$Sign($sk_I, refund||nT||nwCom_R$)
  Relay.Send(nrt_R)
procedure RELAY
  $nrt_R \leftarrow$Intermediary.Receive()
  if $\neg$Verify($pk_I, refund||nT||nwCom_R, nrt_R$) $= 1$ **then**
    Abort() ▷ invalid refund token
  $ncsk_R \leftarrow (nwpk_R, nwsk_R, \perp)$ ▷ match client format
  $nS_R \leftarrow (0, hc^0)$

Algorithm 4 Nano-Pay Protocol between the client and relay to forward a single nanopayment. Run periodically throughout the lifetime of the circuit.

procedure CLIENT($nT, ncsk_C, nS_C$)
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  parse $ncsk_C$ as $(nwpk_C, nwsk_C, HC)$
  parse nS_C as (k, hc^k)
  if $k >= n$ **then**
    Abort() ▷ out of nanopayments, setup a new channel
  $nS_C \leftarrow (k+1, HC[k+1])$
  Relay.Send($HC[k+1]$)
procedure RELAY(nT, nS_R)
  $hc^{k+1} \leftarrow$Client.Receive()
  parse nS_R as (k, hs^k)
  if $k+1 >= n \vee Hash(hc^{k+1}) \neq hc^k$ **then**
    Abort() ▷ invalid nanopayment
  $nS_R \leftarrow (hs^{k+1}, k+1)$

C. Algorithms

APPENDIX
FORMAL DEFINITIONS AND PROOFS

A. Definitions

Anonymity and Security for nanopayment channel: Our objective in this work is to provide an efficient, secure and privacy-preserving payment system for Tor network bandwidth. Our nanopayment channel is built on the top of an existing micropayment channel as designed by Green and Miers [16]. Intuitively, the Pay protocol of their bidirec-

Algorithm 5 Nano-Close Protocol between an end user (client or relay) and an intermediary to close out the nanopayment channel and receive a micropayment wallet. Run any time after the circuit closure. Also, the relay must close first

$\forall E \in \{Client, Relay\}$
procedure ENDUSER($pp, pk_I, w_E, nT, ncsk_E, nS_E$)
  parse w_E as $(B_E, wpk_E, wsk_E, r, \sigma_E^w)$
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  parse $ncsk_E$ as $(nwpk_E, nwsk_E, \_)$
  parse nS_E as (k, hc^k)
  $\epsilon_E \leftarrow \delta_C * k$ if (EndUser = Client) else $\delta_R * k$
  $(wpk'_E, wsk'_E, wCom'_E, \pi'_E) \leftarrow$Wal($pp, pk_I, wpk_B, \sigma_E^w, B_E, \epsilon_E$)
  Intermediary.Send($wpk_E, wCom'_E, \pi'_E, nT, \epsilon_E, k, hc^k$)
procedure INTERMEDIARY(pp, sk_I, S_I, nS_I)
  $(wpk_E, wCom'_E, \pi_E, nT, \epsilon_E, k, hc^k) \leftarrow$EndUser.Receive()
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  if $\epsilon_E < 0 \wedge closed \notin nS_I[nT]$ **then**
    Abort() ▷ client attempting to close before relay
  if $\neg$Verify(π_E) $\vee\ nS_I[nT] \neq established$ **then**
    Abort() ▷ invalid wallet or channel
  if $k > n \vee \neg$VerifyHC(hc^0, k, hc^k) **then**
    Abort() ▷ invalid payment hash chain
  $nS_I[nT] \leftarrow closed||hc^k$
  $rt'_E \leftarrow$Sign($sk_I, refund||wCom'_E$)
  EndUser.Send(rt'_E)
procedure ENDUSER
  $rt'_E \leftarrow$Intermediary.Receive()
  if $\neg$Verify($pk_I, refund||wCom'_E, rt'_E$) $= 1$ **then**
    Abort() ▷ invalid refund token
  parse $ncsk_E$ as $(nwpk_E, nwsk_E, \perp)$
  $\sigma_E^{rev(nrt)} \leftarrow$Sign($nwsk_E, revoke||nwpk_E$)
  Intermediary.Send($nwpk_E, \sigma^{rev(nrt)}$)
procedure INTERMEDIARY
  $(nwpk_E, \sigma_E^{rev(nrt)}) \leftarrow$EndUser.Receive()
  if $nwpk_E \in S_I \vee \neg$Verify($nwpk_E, \sigma^{rev(nrt)}$) **then**
    Abort() ▷ unregistered channel or revocation token
  $S_I[nwpk_E] \leftarrow \sigma^{rev(nrt)}$
  EndUser.Send($verified$)
procedure ENDUSER
  $ver \leftarrow$Intermediary.Receive()
  $w'_E \leftarrow$Intermediary.Blindsig($ver, wpk'_E||B_E + \epsilon_E$)

Algorithm 6 Nano-Refund Algorithm by an end user to close a micropayment channel and claim ledger funds. This is a modified version of Bolt's Refund algorithm to also allows for granular claims on open nanopayment channels

$\forall E \in \{Client, Relay\}$
function ENDUSER($pp, csk_E, w_E, nT, ncsk_E, nS_E, nrt_E$)
  parse csk_E as $(\_, sk_E, \_, \_, \_)$
  parse w_E as $(B_E, \_, \_, \_, \_)$
  parse nT as $(\delta_C, \delta_B, \_, \_)$
  parse $ncsk_E$ as $(nwpk_E, \_, \_)$
  parse nS_E as (k, hc^k)
  $\delta_E \leftarrow \delta_C$ if (EndUser = Client) else δ_R
  $m_E \leftarrow (refund||nT||nwpk_E||B_E + \delta_E * n, nrt_E, hc_E^k, k_E)$
  $nrc_E \leftarrow (m_E, Sign(sk_E, m_E))$
  return nrc_E

Algorithm 7 Nano-Refute Algorithm by an intermediary to respond to an end user's refund claim by posting its own channel closure message to the ledger

$\forall E \in \{Client, Relay\}$
function INTERMEDIARY($pp, T_E, S_I, nS_I, nrc_E$)
  parse nrc_E as (m_E, σ_E^m)
  parse m_E as $(refund||nT||nwpk_E||B_E^{full}, nrt_E, k_E, hc_E^k)$
  ▷ $B_E^{full} \leftarrow$ balance if nanopayment channel were saturated
  parse T_E as $(pk_E, \_)$
  if $\neg$Verify(pk_E, m_E, σ_E^m) **then**
    Abort() ▷ bad signature, well be rejeced by ledger
  if $\neg$Verify$(pk_I, (refund||nT||nwpk_E||B_E^{full}), nrt_E)$ **then**
    Abort() ▷ unapproved refund token
  if $S_I[nwpk_E] \neq \perp$ **then**
    ▷ E is posting an old token, I should refute
    $\sigma_E^{rev(nrt)} \leftarrow S_I[nwpk_E]$
    $nrc_I \leftarrow ((revoked, \sigma_E^{rev(nrt)}), Sign((revoked, \sigma^{rev(nrt)})))$
  ▷ Everything checks out; accept the closure
  $hc^k \leftarrow nS_I[nT]$
  $nrc_I \leftarrow ((accepted, k_I, hc_I^k), Sign(accepted, k_I, hc_E^k))$
  return nrc_I

Algorithm 8 Nano-Resolve Algorithm run by the ledger (and everyone verifying the ledger) to resolve all channel closure messages and allocate the appropriate final balances

▷ Returns the tuple $(B_E^{final}, B_I^{final})$
function LEDGER($pp, T_E, T_I, nrc_E, nrc_I$)
  $B^{total} = B_E^{init} + B_I^{init}$
  parse nrc_E as (m_E, σ_E^m)
  parse nrc_I as (m_I, σ_I^m)
  parse m_E as $(refund||nT||nwpk_E||B_E^{full}, nrt_E, k_E, hc_E^k)$
  ▷ $B_E^{full} \leftarrow$ balance if nanopayment channel were saturated
  parse nT as $(\delta_C, \delta_R, n, hc^0)$
  $\delta_E \leftarrow \delta_C$ if (EndUser = Client) else δ_R
  if $nrc_E = \perp$ **then**
    ▷ E failed to respond closure request in time
    return $(0, B_{total})$
  if $\neg$Verify$(pk_E, m_E, \sigma_E^m) \vee \neg$Verify$(pk_I, m_I, \sigma_I^m)$ **then**
    return $\perp$ ▷ messages could not be authenticated
  if $\neg$Verify$(pk_I, refund||nT||nwpk_E||B_E^{full}, nrt_E)$ **then**
    return $(0, B_{total})$ ▷ E is attempting to use invalid token
  if $revoked \in m_I$ **then**
    parse m_I as $(revoked, \sigma_E^{rev(nrt)})$
    if Verify$(nwpk_E, \sigma_E^{rev(nrt)})$ **then**
      return $(0, B_{total})$ ▷ E is trying to use old channel
    else
      return $(B_{total}, 0)$ ▷ invalid revocation from I
  ▷ micropayments settled, now resolve nanopayments
  parse m_I as $(accepted, k_I, hc_I^0)$
  if $k_I \leq k_E \leq n \wedge$VerifyHC$(hc^0, k_E, hc_E^k))$ **then**
    ▷ E has the highest hash preimage
    $B_E^{final} = B_E^{full} - \delta_E * (n - k_E)$
    $B_I^{final} = B_{total} - B_E^{full} + \delta_E * (n - k_E)$
  if $k_E \leq k_I \leq n \wedge$VerifyHC$(hc^0, k_I, hc_I^k))$ **then**
    ▷ I has the highest hash preimage
    $B_E^{final} = B_E^{full} - \delta_E * (n - k_I)$
    $B_I^{final} = B_{total} - B_E^{full} + \delta_E * (n - k_I)$
  return $(B_E^{final}, B_I^{final})$

tional channel is replaced by our set of Nano-Setup, Nano-Establish, Nano-Pay and Nano-Close protocols which allows high-granularity payments of up to n iterations at the cost of roughly two Pay protocols. We require that the intermediary does not learn more than the number of nanopayments realized between an unknown Tor client and a unknown relay. [12] Moreover, we require that the nanopayment protocol always produce a correct outcome for each valid execution of the protocol. Informally, the anonymity guarantees provided by the nanopayment channel states that any relay (except the guard relay) of a circuit learns no information except that a valid nanopayment channel establishment, payment or closure has occurred over an open micropayment channel with some intermediary. A particular relay should not be able to link any two nanopayment channel for separate circuits that it operates.

We reuse the payment anonymity and balance properties of Green and Miers [43] for an Anonymous Payment Channel (APC scheme) but we adapt them for our tripartite protocol. The scheme requires a privacy property that holds against the intermediary, a privacy property that holds against a relay, and a balance property to define monetary security. We prove that if there exists an adversary able to break the anonymity property, then this adversary is able to distinguish the Real experiment from the Ideal experiment of an APC scheme with non-negligible advantage. Furthermore, we prove that the only adversary able to break the balance property is an adversary able to break preliminary security assumptions.

1) Payment anonymity with respect to the intermediary:: Let $\mathcal{A}$ be an adversary playing the role of intermediary. We consider an experiment involving P customers (a.k.a. Tor client) and Q relays, each interacting with the intermediary as follows. First, $\mathcal{A}$ is given pp, then outputs $T_{\mathcal{M}}$. Next $\mathcal{A}$ issues the following queries in any order:

Initialize channel for $\mathcal{C}_i$ and $\mathcal{R}_j$. When $\mathcal{A}$ makes this query on input B^{cust}, B^{inter}, it obtains the commitment $T_{\mathcal{C}}^i$ generated as

$$(T_{\mathcal{C}}^i, csk_{\mathcal{C}}^i) \leftarrow_{\$} Init_{\mathcal{C}}(pp, B^{cust}, B^{inter})$$

where the customer might also be a relay. In this case, the intermediary obtains the commitment $T_{\mathcal{R}}^j$ generated as

$$(T_{\mathcal{R}}^j, csk_{\mathcal{R}}^j) \leftarrow_{\$} Init_{\mathcal{R}}(pp, B^{relay}, B^{inter})$$

Establish channel with $\mathcal{C}_i$ and $\mathcal{R}_j$. In this query, $\mathcal{A}$ executes the Establish protocol with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$) as

$$Establish(\{\mathcal{C}(pp, T_{\mathcal{M}}, csk_{\mathcal{C}}^i)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary's state. We denote the customer's output as w_i, where w_i may be $\perp$.

Nano-Setup from $\mathcal{C}_i$. In this query, if $w_i \neq \perp$, then $\mathcal{A}$ executes the Nano-Setup to escrow ϵ with $\mathcal{C}_i$ as:

$$\text{Nano-Setup}(\{\mathcal{C}(pp, \epsilon, w_{\mathcal{C}}^i)\}, \{\mathcal{A}(state)\})$$

[12] Due to the fact that moneTor nanopayment channels are inherently transparently, we do not require that Nano-Setup and Nano-Establish protocols are unlinkable to the Nano-Close protocol from the perspective of the relay and the intermediary.

Where $state$ is the adversary's state. We denote the customer's output as $w^i_{\mathcal{C}}$, the hashchain root hc^0, the customer's nanopayment secret $ncsk_{\mathcal{C}}$, the customer's state $nS_{\mathcal{C}}$ and the refund token $nrt_{\mathcal{C}}$, where any may be $\perp$.

Nano-Establish from $\mathcal{R}_j$. In this query, if $w^j_{\mathcal{R}}$ and $nT \neq \perp$, then $\mathcal{A}$ executes the Nano-Establish to register the nanopayment channel with the relay $\mathcal{R}_j$ as:

$$\text{Nano-Establish}(\{\mathcal{R}(pp, w^j_{\mathcal{R}}, nT)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary state. We denote the relay's output as $w^j_{\mathcal{R}}$, the refund token $nrt_{\mathcal{R}}$, the relay's nanopayment secret $ncsk_{\mathcal{R}}$ and the state of the relay's nanopayment channel $nS_{\mathcal{R}}$.

Nano-Close from $\mathcal{C}_i$ and $\mathcal{R}_j$. In this query, if $\epsilon^i_{\mathcal{C}}$, nT, $ncsk_{\mathcal{C}}$ and $nS_{\mathcal{C}} \neq \perp$, then $\mathcal{A}$ executes the Nano-Close to close the nanopayment channel and update the micropayment wallet with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$).

$$\text{Nano-Close}(\{\mathcal{C}(pp, \epsilon^i_{\mathcal{C}}, nT, ncsk_{\mathcal{C}}, nS_{\mathcal{C}})\}, \{\mathcal{A}(state)\}) \rightarrow w^i_{\mathcal{C}}$$

Where $state$ is the adversary's state. We denote the customer's and relay's output as $w^i_{\mathcal{C}}$ (resp. $w^j_{\mathcal{R}}$), where it may be $\perp$.

Finalize with $\mathcal{C}_i$ (resp. $\mathcal{R}_j$). When $\mathcal{A}$ makes this query, it obtains $rc^i_{\mathcal{C}}$, computed as $rc_{\mathcal{C}} \leftarrow\!\!\$\ Refund(pp, w^i_{\mathcal{C}})$.

We say that $\mathcal{A}$ is $legal$ if $\mathcal{A}$ never asks to spend from a wallet where $w^i_{\mathcal{C}}$ or $w^j_{\mathcal{R}}$ is $\perp$ or undefined, and where $\mathcal{A}$ never asks C_i to spend unless the customer has sufficient balance to complete the spend.

Let pp' be an auxiliary trapdoor not available to the participants of the real protocol. We require the existence of a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that for all $T_{\mathcal{M}}$, no allowed adversary $\mathcal{A}$ can distinguish the following two experiments with non-negligible advantage:

Real experiment. In this experiment, all responses are computed as described in our Algorithms.

Ideal experiment. In this experiment, the micropayment operations are handled using the procedure above. However, for the nanopayment procedures, $\mathcal{A}$ does not interact with $\mathcal{C}_i$ and $\mathcal{R}_j$ but instead interacts with a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$.

2) Payment anonymity with respect to the relay.: Let $\mathcal{A}$ be an adversary playing the role of relay. We consider an experiment involving P customers (a.k.a. Tor clients), each interacting with the relay as follows. First, $\mathcal{A}$ establishes a micropayment channel with the intermediary. Next, $\mathcal{A}$ issues the following queries in any order:

Nano-Establish from $\mathcal{C}_i$. In this query, nT may be $\perp$, then $\mathcal{A}$ executes only the part of Nano-Establish which interacts with $\mathcal{C}_i$:

$$\text{Nano-Establish}(\{\mathcal{C}(pp, nT)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary state. We denote the customer's output nT, which may be $\perp$.

Nano-Pay from $\mathcal{C}_i$. In this query, $nT \neq \perp$ and p_k may be $\perp$, then $\mathcal{A}$ executes the Nano-Pay protocol for an amount δ with $\mathcal{C}_i$ as:

$$\text{Nano-Pay}(\{\mathcal{C}(pp, \delta, p_k)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary's state and p_k is the preimage of the current hash stored in the adversary's state, or $\perp$.

We say that $\mathcal{A}$ is $legal$ if $\mathcal{A}$ never asks to spend more than $n * \delta$.

Let pp' be an auxiliary trapdoor not available to the participants of the real protocol. We require the existence of a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that for all $T_{\mathcal{M}}$, no allowed adversary $\mathcal{A}$ can distinguish the following two experiments with non-negligible advantage:

Real experiment. In this experiment, all responses are computed as described in our Algorithms.

Ideal experiment. In this experiment, the micropayment operations and nanopayment operations with the intermediary are handled using our algorithms. However, for the nanopayment procedures between the Tor client and the adversary relay, $\mathcal{A}$ does not interact with $\mathcal{C}_i$ but instead interacts with $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$.

3) Payment Security (Balance): Let $\mathcal{A}$ an adversary playing the role of relay. We consider an experiment involving P honest Tor clients $\mathcal{C}_1, ..., \mathcal{C}_P$ interacting with the relay. We assume the micropayment channels have been properly setup and established with the intermediary and that the intermediary continues to interact honestly with the client and relay.

Given the micropayment channel setup and established, parties hold funds valued at B^{cust} and $B^{\mathcal{A}}$. Let $bal_{\mathcal{A}} \leftarrow 0$ be the amount of funds the adversary may claim. Now $\mathcal{A}$ may issue the following queries in any order:

Nano-Establish from $\mathcal{C}_i$. In this query, nT may be $\perp$, then $\mathcal{A}$ executes only the part of Nano-Establish which interacts with $\mathcal{C}_i$:

$$\text{Nano-Establish}(\{\mathcal{C}(pp, nT)\}, \{\mathcal{A}(state)\})$$

Where $state$ is the adversary state. The adversary obtains nT and establishes the nanopayment channel with the intermediary.

Nano-Pay from $\mathcal{C}_i$. The nanopayment channel has been correctly established before. This query can executed up to n times before **Nano-close** is called. For each execution, $nT \neq \perp$ and p_k may be $\perp$. $\mathcal{A}$ executes the Nano-Pay protocol for an amount δ with $\mathcal{C}_i$ as:

$$\text{Nano-Pay}(\{\mathcal{C}(pp, \delta, p_k)\}, \{\mathcal{A}(state)\}) \rightarrow p_k$$

If $H(p_k)$ matches the hash stored in the adversary's state, then $bal_{\mathcal{A}} = bal_{\mathcal{A}} + \delta$ and $H(p_k)$ is stored in the internal state. If it does not match, we output $\perp$.

Nano-Close with intermediary. In this query, $\epsilon_{\mathcal{A}} \leftarrow k * \delta$ for k Nano-Pay executions. $nT, ncsk_{\mathcal{A}}, nS_{\mathcal{A}} \neq \perp$, then $\mathcal{A}$ executes the Nano-Close protocol to close its leg of the nanopayment channel and claim funds to the intermediary.

$$Nanoclose(\{\mathcal{A}(pp, \epsilon^i_{\mathcal{A}}, nT, ncsk_{\mathcal{A}}, nS_{\mathcal{A}})\},$$

$$\{Intermediary(state)\}) \rightarrow w^i_{\mathcal{A}}$$

We denote the adversary output $w^i_{\mathcal{A}}$, where it may be $\perp$. The Tor client closes also its leg of the nanopayment channel with the intermediary to transfer $k * \delta$ and update its wallet accordingly. At any point, all parties have the option to call Nano-Refund to initiate a partial or full refund of their escrowed fund and close the nanopayment channel. We say that $\mathcal{A}$ is $legal$ if it never agrees to execute the Nano-Pay protocol upon $nT = \perp$. We further restrict $\mathcal{A}$ to establish one nanopayment channel per micropayment channel established with any Tor client. $\mathcal{A}$ wins if after executions of queries, $bal_{\mathcal{A}} > k * \delta$.

B. Theorem

The nanopayment channel scheme satisfies the properties of anonymity (A1, A2) and security (A3) under the restriction that the adversary does not abort before Nano-Close finished, the restrictions that at most one nanopayment channel can be open per micropayment channel, the assumptions that the commitment scheme is secure, the zero-knowledge system is simulation extractable and zero-knowledge, and the hash function used to create the hashchain and verify the preimage during the Nano-Pay is a cryptographic hash function.

C. Proofs

1) Anonymity: We prove that the nanopayment channel scheme satisfies our anonymity properties using a simulator $\mathcal{S}^{X-Y(\cdot)}(pp, pp', \cdot)$ such that no allowed adversary $\mathcal{A}$ can distinguish the Real experiment from the Ideal experiment with non-negligible advantage. The way this proof proceeds requires honest runs of the appropriate algorithms for the micropayment channel. When Nanopayment channel operations are called, the client side or relay side of the protocol is emulated by the simulator for the Ideal experiment. To prove that they are indistinguishable, we borrow Green and Miers's proof and extend it to our notion of payment anonymity to the intermediary, and to the relay. We start with the Real experiment and we create Games which modify elements of the protocol until we match the Ideal experiment conducted with the simulator $\mathcal{S}$. When $\mathcal{A}$ calls the simulator $\mathcal{S}$ on legal interactions, the simulator emulates the Tor client or relay part of the protocol, depending on which step of the protocol we perform.

Let be ν_1, ν_2 be negligible functions and let **Adv[Game i]** be $\mathcal{A}$'s advantage in distinguishing the output of **Game i** from the Real Distribution.

Game 0. This is the Real experiment: Nano-Setup, Nano-Establish and Nano-Close between customers (Tor clients) and the intermediary.

Game 1. This game is identical to **Game 0** except that we replace NIZK proofs generated by the customer at the Nano-Setup and Nano-Close with simulated proofs (we assume the existence of a ZK simulation algorithm which can extract a simulated proof). If the proof system is zero-knowledge, then **Adv[Game 1]** $\leq \nu_1$.

Game 2. This game is identical to **Game 1** except that we replace the commitments $nwCom_C$, $nwCom_R$, $wCom'_C$ and $wCom'_R$ by commitments on random messages. If the commitment scheme is computationally hiding, then **Adv[Game 2]** $-$ **Adv[Game 1]** $\leq \nu_2$.

Game 3. This game is identical to **Game 2** except that we replace the root of the hashchain $HC[0]$ by a value generated from Random(). Note that Random() was also used for the original value, therefore **Adv[Game 3]** $-$ **Adv[Game 2]** $= 0$.

Game 4. This game is identical to **Game 3** except that we replace $wpk_C, nwpk_C, wpk_R, nwpk_R$ with random keys using the KeyGen algorithm described for anonymous micropayment channels. Since the distribution is identical to the distribution of original values, **Adv[Game 4]** $-$ **Adv[Game 2]** $= 0$

We have started with the Real experiment and modified elements of the protocols from a series of Games to come up with a computationally indistinguishable experiment conducted by $\mathcal{S}$ from the Real experiment. Since $\mathcal{A}$ cannot distinguish the real experiment from the Ideal experiment obtained in **Game 4.** with non-negligible advantage, the interaction between customers and intermediary is anonymous.

Now, we have to prove the indistinguishably between the Real experiment and the Ideal experiment for the payment anonymity property with the relay. We proceed with the same logic:

Game 0'. This is the Real experiment: Nano-Establish and Nano-Pay between Tor clients and relays.

Game 1'. This game is identical to **Game 0'** except that we replace the root of the hashchain $HC[0]$ by a value generated from Random() in the Nano-Establish interaction. Note that Random() was also used for the original value, therefore **Adv[Game 1']** $-$ **Adv[Game 0']** $= 0$

Game 2'. This game is identical to **Game 1'** except that we replace the preimage p_k sent to the relay by a value generated from Random(). In the random oracle model, both original value and simulated one provide from the same distribution, hence **Adv[Game 2']** $-$ **Adv[Game 1']** $= 0$

Since **Game 2'** is identical in the Ideal experiment, the interaction between Tor clients and relays is anonymous.

By showing that the interaction with the intermediary and the interaction with the relay through the nanopayment algorithms are anonymous, we conclude that our nanopayment channel is anonymous.

2) Security (Balance): We prove that the Nanopayment channel satisfies the security definition if the micropayment channel is itself secure, if the hash function used is a cryptographic hash function (i.e. it behaves like a random function, it's easy to compute a hash of any given data, it's computationally hard to find a valid preimage of a given hash value, and it is unlikely to find two different pieces of data that hash to the same value), and if the signature scheme is EU-CMA secure (i.e. Existential Unforgeability under a Chosen Message Attack).

To win, $\mathcal{A}$ must claim more money than the agreed upon price between a honest client and the adversary. The adversary must make this claim while running a protocol that is indistinguishable from the honest protocol. The Nano-Setup

protocol borrows the same structure as the provably secure Pay protocol. This includes the soundness of the zero-knowledge proof, the binding property of the commitment scheme and the unforgeability of the signature scheme. At this step, the adversary cannot win against the Tor client and claim more than $k * \delta$ where k is 0 since no Nano-Pay has been executed yet. For the following steps of the protocol, we proceed by showing that $\mathcal{A}$ cannot diverge from the protocol and claim more than $k * \delta$ with our classical security assumptions. If $\mathcal{A}$ could succeed this game, it would mean that there exist an indistinguishable experiment from the Real experiment where $\mathcal{A}$ ends up with more than $k * \delta$.

Game 0. This is the Real experiment.

Game 1. This game is identical to Game 0 except that we replace hc^0 in nT by a value chosen by $\mathcal{A}$ from a hashchain created by $\mathcal{A}$. From this hashchain, $\mathcal{A}$ creates nT'. If the intermediary is honest, the nanopayment cannot be established because nT' is unknown to the intermediary for this micropayment channel. If the intermediary is dishonest, then it can accept nT' but cannot prove, under the assumption that the signature scheme is unforgeable in the usual sense (EU-CMA secure), that the client holds a refund token with nT' instead of nT. Hence, **Adv[Game 1]** $\leq \nu_1$.

Game 2. This game is identical to Game 1 except that $\mathcal{A}$ tries in the Nano-Pay protocol to find herself a preimage to the stored hashchain, and claim more than δ. Assuming the hash function is a cryptographic hash function, then the adversary cannot find a preimage unless the Tor client sends it to issue a payment. Hence, **Adv[Game 2]** $\leq \nu_2$.

Finally, the Nano-Close protocol borrows the micropayment Pay protocol to update the micropayment wallet according to the number k of preimages the adversary received from the Tor client. The Pay protocol has been proved secure by Green and Miers, hence we observe that the adversary cannot win the game with a non-negligible probability (claim more than $k * \delta$).

APPENDIX
SCHEDULING

A. background: scheduling

Tor handles multiple queues of cells for each circuit and writes cells in the outbound connection while favoring bursty over bulky traffic. The main idea is to prioritize circuit handling interactive data streams, like chats or web browsing. Tor uses an heuristic called EWMA [44] (i.e., computes the exponentially weighted moving average for the number of cells sent on each circuit) to decide which circuit to prioritize. Recently, the efficiency of EWMA has been improved with the Kist [7] scheduler used to reduce the congestion on the kernel outbound queue and push back this delicate problem on to the Tor layer.

In an ideal network, we might expect that traffic movement is an exclusive function of the raw bandwidth capacity in each edge connection and the scheduling algorithm implemented at each node. The Tor network employs EWMA to favor interactive web clients over continuous bulk clients. In moneTor, we originally proposed to modify EWMA with a simple linear scaling factor that would favor paid circuits.

$$A_{t+\Delta t} = A_t \times 0.5^{\Delta t/H} \tag{3}$$

$$A'_{t+\Delta t} = A_{t+\Delta t}/\beta + C_{t,t+\Delta t} \tag{4}$$

Defined in Tang and Goldeberg's original paper, A is a variable score used to sort circuits such that the circuit with the lowest A is always next on the scheduling queue. C is the number of cells relayed within Δt, the time since the previous observation, and H is a global representing the half-life decay interval in the score. Our added term, $\beta \in [1, \text{inf})$, is a tunable parameter such that $Bandwidth_{premium} = Bandwidth_{nonpremium} \times \beta$ for any given circuit under ideal conditions.

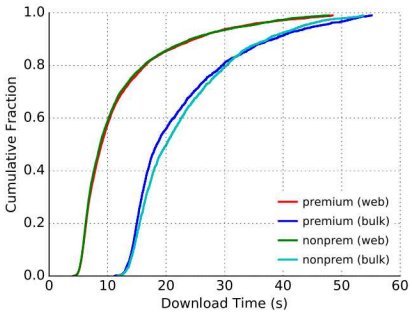


Fig. 9: Prioritized Scheduling — CDF download times for superimposed web and bulk clients where premium status is enforced only via scheduling.

B. Obstacles

Implementation into the concrete Tor infrastructure has proven to be a considerably more complex problem. Upon failing to achieve meaningful differentiation with low values of β, we adopted a more blunt policy which *always* services premium circuits first and implemented it in a zero-overhead version of moneTor.[13] The results are displayed in Figure 9. Although we observed some moderate differentiation, the difference falls well short of the benefit needed to incentivize paid users as well as our expectations for such an inequitable scheduling policy. This result holds even under very heavy levels of induced congestion.

C. Investigation

Our negative results can be explained if scheduling is not the most decisive determining factor in performance. To verify this hypothesis, we studied the incoming queue from which the scheduler is able to select new active circuits. Figure 10 illustrates the temporal load in the queue at a single exit relay over a one-minute time span. The height of the curve represents the total number of cells waiting to be serviced at

[13]This ideal version of moneTor strips away all payment operations and instead passes a single signal through the network to distinguish premium circuits.

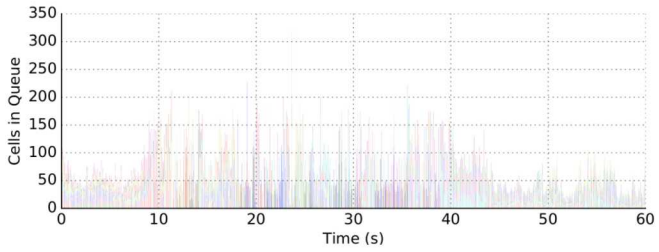


Fig. 10: Queue Temporal Profile (60 seconds) — Size of the scheduling buffer over time at a single exit relay in terms of number of cells. Colors group cells belonging to the same circuit.

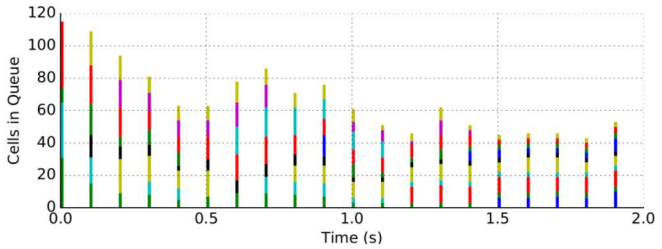


Fig. 11: Queue Temporal Profile (2 seconds) — Size of the scheduling buffer over time at a single exit relay in terms of number of cells. Colors group cells belonging to the same circuit.

each continuous point in time while the colors group quantities of cells that belong to the same circuit. Figure 11 displays a subset of the same information within a smaller time interval.[14] In the aforementioned figures, notice that the queue is only populated for a period of 10 ms before it is completely flushed, implying the queue spends the vast majority of its time empty. This 10 ms window is a product of Tor's internal event handling framework and is consistent with data from Jansen et al [45]. We found in an analysis of the line-by-line observations of the queue activity that while cells are flushed in the correct order, they appear in the queue at roughly equal proportions. In effect, bandwidth in our simulation is not constrained by the ordering policy of the scheduler but rather by the rate at which they arrive from the network.

D. Discussion

There are two plausible contributing explanations. First, it may be the case that other network control mechanisms within the Tor codebase constrain the flow of cells, rendering the mostly idle scheduler to be ineffective. This may be caused by any combination of factors including point-to-point flow control, connection throttling, or some less documented threshold embedded in the code. The positive results covered in Section VIII-C3 suggests that flow-control may play a key role. The second explanation states that even in high-congestion, the constraining network bottleneck does not lie within the Tor network itself but at the exit relay interface with external servers on the web. In this scenario, cell queuing within Tor is not nearly as important as the TCP/IP packet handling at each exit relay.

We must emphasize that our results are in no way indicative or predictive of the state of affairs on the live Tor network. The aforementioned experiments were performed on a considerably smaller scale with simplistic models for network topology and user behavior. What can be said is that networking as a whole is an immensely complex and unpredictable domain and that the attainment of a simulation environment conducive to effective scheduling is, at the very least, nontrivial.

[14] While the graph has the visual appearance of a bar graph, this is just a function of the striking data pattern. In actuality, the plot displays a stacked area graph.