OpenTor: Anonymity as a Commodity Service

Elli Androulaki, Mariana Raykova, Angelos Stavrou, and Steven Bellovin

Columbia University
{elli,mariana,angel,smb}@cs.columbia.edu

Technical Report CUCS-031-07

Abstract. Despite the growth of the Internet and the increasing concern for privacy of online communications, current deployments of anonymization networks depends on a very small set of nodes that volunteer their bandwidth. We believe that the main reason is not disbelief in their ability to protect anonymity, but rather the practical limitations in bandwidth and latency that stem from limited participation. This limited participation, in turn, is due to a lack of incentives. We propose providing economic incentives, which historically have worked very well. In this technical report, we demonstrate a payment scheme that can be used to compensate nodes which provide anonymity in Tor, an existing onion routing, anonymizing network. We show that current anonymous payment schemes are not suitable and introduce a hybrid payment system based on a combination of the Peppercoin Micropayment system and a new type of "one use" electronic cash. Our system claims to maintain users' anonymity, although payment techniques mentioned previously — when adopted individually — provably fail.

1 Introduction

Anonymous networking has been with us since 1981 [2]. A more practical scheme, Onion Routing, was first described in 1995 [5]. Despite that, there is little practical use.

Some of the problem is undoubtedly sociological: most people do not feel the need to protect their privacy that way. This is one reason that companies such as Zero Knowledge Systems failed. Another problem, though, is that strong anonymity against traffic analysis requires cooperation by many different parties. Any single entity, no matter how trustworthy it appears, can be subverted, whether by technical means, corrupt personnel, or subpoena attacks. All known solutions require routing through multiple parties.

This, though, introduces another problem: economic incentives. In a single-provider anonymity scheme, that problem is conceptually simple: the party desiring privacy pays a privacy provider. This payment can be protected by digital cash [1]. In a multi-provider Mixnet or onion routing network, the problem is more complex, since each party must be paid.

In this paper we introduce a payment system for onion routing networks. The goal is to create incentives for the participants in the network to act in a cooperating manner based on their personal interest.

Any solution must be sound in several dimensions. First, of course, it must protect privacy. This is not a trivial ; witness the many (partial) attacks on various anonymous networking protocols [10, 7]. That said, we do not claim to have fixed those problems. Rather, our goal is to do no harm. In other words, our scheme must not introduce any new vulnerabilities.

Second, we want a system that is in principle deployable. That is, though we assume such things as anonymous payment systems, we do not assume (for example) incorruptible banks. More importantly, we want a system that is compatible with known economic behavior. Thus, while our system assumes that people are willing to pay for privacy, we want a system where customer payment — and forwarding node profits — are related to privacy desired and effort expended. In other words, there must be a profit motive and the opportunity for market forces to work. We also provide mechanisms to detect cheaters — parties who accept payment but don't provide services.

Third, we do not attempt to achieve absolute financial security. Rather, we are willing to accept small amounts of cheating, by senders or forwarders, as long as the amount is bounded and is limited (possibly with some trade-off) by the party who is exposed to loss.

Finally, we want a system that is acceptably efficient. While we do not define this precisely, we do attempt to minimize the number of computationally expensive operations.

Existing digital cash schemes do not provide the necessary features for such a scheme. For example, in Chaum's original scheme a double-spender's identity is exposed. This is perfectly acceptable — that is a form of cheating that should be punished — but in the context of an onion routing network, detecting double spending gives an adversary clues to path setup. Accordingly, we use a hybrid scheme. Our payment systems combine features from the Peppercoin micropayment system [11] and a new type of electronic cash.

Finally, we show that there is a limit to the path length in an onion routing network, if we assume that some nodes are in fact traitors. This in turn bounds the cost of our scheme.

2 System Architecture

2.1 Introduction

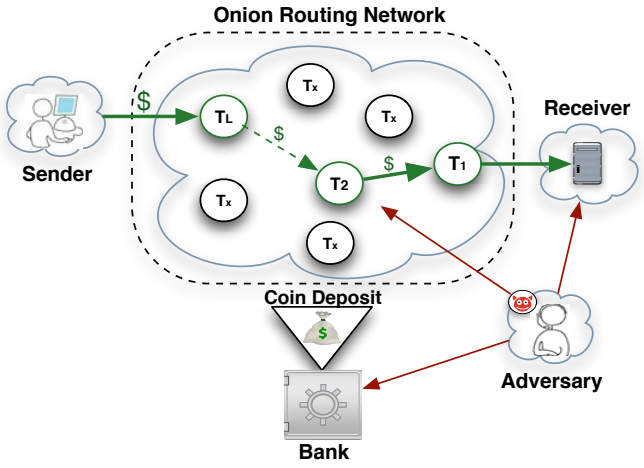

Fig. 1. The OpenTor architecture combines an onion routing anonymity network (TOR) with a payment scheme. Each node $T_1, T_2, T_3, \cdots T_L$, where L is the path length, in the path from the sender to the receiver receives payment in coins for its service. The TOR nodes offering their indirection services are paid by the Bank when they deposit their coins. An adversary can subvert or obtain information from a combination of entities in the anonymity network, including the Bank.

D. Chaum was the first to introduce schemes for anonymous communications networks [2]. Since then, there has been a lot of approaches that use Mix networks [9, 6, ?,?] . However, the use of Mixnets introduces a high end-to-end latency overhead that renders them impractical for most Internet applications including web surfing and Voice over IP (VoIP). To lower the latency overhead and to meet the demand for high-bandwidth communications, anonymity using onion routing was studied [5, ?]. Tor [4] is a second generation onion routing anonymity network that is geared towards protecting the anonymity of generic TCP streams.

2.2 Tor

In Tor, the network consists of a set of nodes that act as traffic indirection points. The region in the dotted lines in Figure 1 shows a typical communication in Tor. To establish communication, the sender sets up a

circuit by selecting a randomly chosen sequence of Tor nodes to form a path to the receiver. In Figure 1 the sender selects nodes $T_1, T_2, T_3, \cdots T_L$, where L is the path length. Initially, it contacts the first node in the path, T_L and negotiates keys using self-signed Diffie-Hellman key exchange [3]. After the key exchange, standard Transport Layer Security (TLS, RFC2246) is used to encrypt the data communication between the sender and T_L and provide forward secrecy. Repeating this process through the established tunnel with T_L, the sender can add more Tor nodes to the path. Each Tor node in the path adds another layer of encryption to the messages exchanged increasing the size of the onion. Each message is decoded partially by a Tor node and the data is forwarded to the next node in the path. When the message reaches the last Tor node on the path, the receiver is revealed to this node and a regular TCP connection is established with the remote site's IP address. This design allows Tor to offer anonymity to virtually any application that runs on top or can be encapsulated by TCP, including IRC, HTTP, SSH and VoIP streams. Lately, there have been efforts to extend the support to UDP traffic. The Tor specification has more detailed information about the low level Tor design `http://tor.eff.org/cvs/doc/tor-spec.txt`

Mixnets were designed assuming a global adversary, i.e., one that could see all links. Tor assumes a more limited adversary, one who can see some arbitrarily-selected links but not all.

2.3 Evaluation of Current Payment Schemes

Currently, Tor consists of a small list (< 100) of public nodes that volunteer their time and bandwidth acting as forwarding points. Having a small set of indirection points limits the number of local Tor nodes, thus increasing the end-to-end system latency. Furthermore, it is much easier and more effective for an adversary to perform network traffic, timing, and infiltration attacks. We posit that if there were an economic/monetary incentive for nodes to participate, all parties can benefit: senders can achieve better network performance and higher degree of anonymity, and Tor nodes can be paid for the offered service. However, to prevent nodes from cheating and to form a network which is robust in the long run, Tor nodes must have an incentive which is related to the amount of traffic they forward. In addition, participating nodes should have strong identities which are related to their real identities rather than to their network IP address. Strong identities enables us to enforce a payment protocol which adheres to Authentication, Authorization and Accountability (AAA) principles, leading to a network robust anonymity network. For the rest of the paper, we will assume that the payment scheme provide us with strong identities; we thus don't try to address attacks that stem from forging of identities (i.e. Sybil attacks).

Before we start the analysis of different payment schemes for an anonymizing network, we have to step back and look at what we are actually trying to achieve: hide the communication between senders and receivers. This goal is common among all anonymity networks, including all Mixnets and onion routing-based schemes. There exists two major types of payments schemes that could be potentially be used to pay for anonymity: one that uses micropayment coins[8] and one that uses anonymous coins, that is any kind of e-cash scheme [1]. We will show that both of these schemes, if used throughout, make the anonymity system vulnerable to attacks that can, under some conditions, reveal the anonymity of the senders. In all of the schemes, we will assume that there is a payment clearance entity, the Bank, and that the sender has to provide payment for all of the nodes $T_1, T_2, T_3, \cdots T_L$ that forward the sender's traffic to the receiver (see Figure 1).

Identity-bound Payment Schemes A very broad range of payment techniques involve schemes in which the identity of the user is directly or indirectly connected with the coins exchanged. Micropayments [8],which was designed to be efficient for small, online transactions, is an example of this class of scheme. When used to pay Tor nodes, micropayments provide immediate accountability since misbehavior from any entity can be easily identified. However, in the context of the Tor network this property has adverse implications: upon clearance, the Bank will obtain global knowledge about all transactions in the anonymity network. If the user uses his own coins to pay the nodes in the path, his identity is exposed to them. Thus, any node in the path to the receiver can identify him with the help of the Bank. To make things worse, the last node in the path can expose his anonymity since he also knows the receiver. A potential way to work around this problem is

to distribute the payments in a way that payments are made only to immediate neighbors in the system. Using this payment strategy, the sender gives the first node on the path the payments for all the forwarding nodes on the path. Then each node keeps his appropriate payment and passes the rest to its successor in the path. This approach makes path tracing much harder and leaks less information but it is far from secure: deposits made by the sender to the first Tor node are still available to the Bank. Counting the coins bound to the sender's identity, the Bank can, with high confidence, infer the number of packets communicated to the sender and link the sender to the receiver. Thus, having identity-bound coins reveals too much information, enabling an adversary with access to payment information to break the system's anonymity using simple inference techniques.

Anonymous Payment Schemes Another potential avenue for payment is to use a completely anonymous payment scheme where coins are checked upon deposit to the Bank. This anonymity property seems desirable for Tor networks but it has also its drawbacks. A possible way of cheating with e-cash is double spending. It can be detected at the moment of depositing coins; thus, the most efficient way to prevent this attack is immediate deposit of payments. Unfortunately, this is problematic in Tor networks because depositing coins can leak information about the paths of the connections in the system. The timing of deposits of the nodes in a Tor connection discloses to the Bank the path and an estimate of the number of packets transfered. To avoid revealing information, nodes should make accumulated deposits. Furthermore the deposit accumulation period should be long enough so that sufficiently many connections will have been established and payments accumulated to avoid leaking information. It is easy to see that if the Tor nodes deposit coins in regular time intervals or based on the number of coins they receive, an adversary can game the system by double spending anonymous coins. Indeed, since anonymous coins are not traceable beyond the first Tor node, sending valid coins only to the first node is enough to prevent it from been traced. For the rest of the nodes, the cheater uses double-spent coins, exploiting their deposit strategy by transmitting many packets in a short period of time. Worse yet, this implies a great deal of trust in the Tor nodes by the sender. Will the forwarding node in fact refrain from depositing the coins for a sufficient interval?

In addition, in e-cash [1], we can use anonymous coins that cannot be connected with identity of the payer unless there is double-spending. Unfortunately, there is a very high network overhead for each transaction because there are a lot of messages that need to be exchanged between the depositor and the Bank. This requirement makes e-cash schemes impractical for our system.

Based on the discussion of the two payment methods, we can now formulate the properties that will be desirable for a payment scheme in Tor network. Preserving the anonymity of the network has two aspects: not being able to link senders with receivers in the network, and not being able to trace the exact path that the packets on a particular connection follow from their source to their destination. Accomplishing this when an adversary can monitor the payment system implies that the payments at each node should be accumulated and deposited only after a long-enough periods. Also, the identity of senders who may be outside the Tor network should remain unknown to the Bank.

2.4 System Requirements

As noted, our goal is to provide a way of paying nodes who offer their services in an anonymizing network. We are not concerned with improving the underlying anonymity; rather, we want to ensure that the new operations and entities do not introduce new vulnerabilities or attacks. In particular, we wish to preserve these basic system properties:

Correctness-Robustness This property guarantees that an honest user, when using honest nodes for his transfer, should be able to transmit his packets to their final destination without risking his anonymity or revealing the Receiver. All forwarding nodes will be paid for their services. The open nature of the system, however, requires that we extend Correctness to Robustness. That is, we wish to limit the number of packets that may be sent without proper payment being received.

Sender-Receiver Unlinkability Let S be a user, who may or may not be member of the anonymizing network, who sends a message M anonymously[1] to a user R. This property has two parts:
- under no circumstances (other than a global adversary) can S be linked to R, even with R's cooperation.
- there must be no way that a third party or a collaboration of third parties can reveal the path from S to R.

2.5 Adversarial Model

As in all systems claiming to offer a level of security, a critical point in our system analysis is the adversarial model that we consider. What are the the assumptions we make about attacker's powers, what are the system entities, and in what ways may they misbehave?

For our convenience, we will assume that we have a sender S who wishes to send a message M to a receiver R. Both, S and R can either be members of the underlying anonymizing network or external users. Also, there is a Bank which handles user' accounts. We make the following assumptions:

Bank is "honest but curious". Namely, although Bank is trusted to perform coin validity checks, generating valid coins whenever requested, and to check double-spending properly, we assume that it can collaborate with a user or a group of users in order to disclose the initiator of a communication or to reveal active communication paths.

Users can be either malicious or honest. They can observe any portion of the traffic through them, observe a limited amount of other traffic, manipulate any packet that goes through them and use this information to compromise the system's anonymity requirements.

3 A Hybrid Payment Scheme

In this section we present a detailed description of our payment protocol. As mentioned before, the payment scheme described, although presented through Tor, can be applied to any onion routing system. At the beginning of the current section, we define *MCoin*, the system's monetary unit, while the second part of this section is focused on a detailed description of the messages in the actual payment protocol.

3.1 MCoins

MCoins are the payment means in our system. There are two types of MCoins: *S-mcoins* (Signed microcoins) which can only be used by Tor nodes, and *A-mcoins* (Anonymous coins) which are coins blindly signed by the Bank. A-mcoins are provided to guests or Tor members who wish to use Tor anonymity services. Both types of coins are extensions of the microcoins introduced in [8, 11].

- **S-mcoins(Signed microcoins)**. S-mcoins are generated by Tor members. They are strongly connected to the identity of the node that created them, as well as to the transaction details for which they are used. Not all S-mcoins are depositable. In fact, as usual in micropayment schemes, only a certain fraction $\frac{1}{s}$ of S-mcoins are depositable, while the rest will be void. However, whenever a deposit is made, s points are removed from Spender's account and added to Recipient's account. An S-mcoin

$$SC_{i\to j} = sig_{T_i}\{MC, VR_{SC}, T_j\},$$

  where T_i is the Tor-node who created it (Spender) and T_j the one who deposits it (Recipient), is a signature of T_i on three quantities:
  - $MC = sig_{T_i}\{T\}$, which is a pure micro-coin. This is the part of S-mcoin that depends on transaction details and determines whether the $SC_{i\to j}$ is depositable or not.

[1] "anonymously", here, denotes "using anonymizing network".

- VR_{SC} is a quantity related to the that mcoins receipt, which will be analyzed later.
- T_j.

– **A-mcoins (Anonymous coins).** A-mcoins are generated by the cooperation of the Bank upon users' requests. In this case users may be members of the network as well as outside nodes. A user buys from the Bank a prefixed number n of microcoins. A-mcoins are of the form

$$AC(r) = sig_B^b\{r\},$$

where r a random number generated by the User, and $sig_B^b\{r\}$ the blind signature of the Bank on it. A-mcoins are all payable and subjected to double-spending checks.

It is crucial that there is no way for T_i to find out in advance which of the S-mcoins he signs will become payable. An S-mcoin is considered payable under some criteria concerning MC; the Bank guarantees that exactly $\frac{1}{s}$ portion of each batch are payable.

Both S-mcoins and A-mcoins are very easily generated. However, we have to note that the "micro-coin" notion is extended here so that an "S-mcoin/A-mcoin" can be considered for a validity check if and only if the receiver presents a receipt that corresponds to VR_{SC} part of it.

3.2 Exchange of Messages

Initial Setting We assume that all nodes participating in Tor have generated a public-secret signature key pair (sk_U^s, pk_U^s), by which they are identified in the network. Additionally, each Tor node has acquired a public-secret encryption key pair (sk_U^e, pk_U^e). Bank also generates a blind signature key pair (sk_B^b, pk_B^b) for signing the A-mcoins. For simplicity, from now on we will $\{M\}_K$ to denote the encryption of message M using key K and $sig_U(M)$ to denote the signature of User U on message M. Apart from the signature and encryption key pairs, we assume global knowledge of two hash functions H and H_r. H will be used for integrity purposes, while H_r will be used for mcoin receipts. Furthermore, we assume that after a Tor circuit setup, Sender S and each of the nodes in the path will agree on a shared secret key. In a similar way, successive nodes in a path establish a secret key for every connection going through them. Apart from H and H_r hashes, which are meant to be used by all, there is a variety of other globally known hash functions $H_1, H_2, \ldots, H_m$ where $m \in N$. Assume that node S wants to talk to R through the path $\langle T_{i_\ell}, Ti_{\ell-1}, \ldots, T_{i_1}\rangle$. Inter-node communication for the circuit established is served through the following secret keys: $K_{ST_{i_l}}$, $K_{ST_{i_{\ell-1}}}$, ..., $K_{ST_{i_1}}$. For integrity verifiability purposes, S shares with each node on the path T_j a hash function H_{ST_j}, where $H_{ST_j} \in \{H_i\}_{i=1}^m\}$.

Payment In the cases of A-mcoins and S-mcoins we have different payment protocols. To spend an A-mcoin, giving the coin with the receipt to the other party is enough. However, the same does not hold for S-mcoins. The S-mcoin generation procedure is strongly connected to how it is spent. Assume that S and M users are involved in a transaction. The Payment procedure involves the following steps:

– S generates the receipt pair $(r, H_r(r))$, where r is a random number provided by S.
– S signs the transaction details T, which should include a serial number for the coin, to produce $MC = sig_S(T)$. This part of the mcoin will be used for the payability check during the deposit procedure.
– S signs the MC part of the mcoin to produce the S-mcoin's final form: $SC_{S\to M}(r) = sig_S(MC, H_r(r), M)$.
– S gives this mcoin $SC_{S\to M}(r)$ and the corresponding receipt r to M. In order for the mcoin to be considered for a paybility check, M must show r to the Bank.

Communication Protocol Description Assume that S wants to send R a message M through nodes T_ℓ, ..., T_1. Then ℓ will be the number of coins that S needs to pay to use Tor. S can be either a guest node or an active node in the Tor network.

The main concept in our payment scheme is that S uses ℓ mcoins to pay the first node in the path T_{i_ℓ}. If S is a Tor node, he uses S-mcoins; otherwise, he uses A-mcoins. Subsequently T_{i_ℓ} pays $\ell - 1$ signed mcoins

(S-mcoins) to $T_{i_{\ell-1}}$. In general, a node $T_{i_{\ell-k}}$ in the path pays the next hop node $T_{i_{\ell-(k-1)}}$ $N-(k-1)$ S-mcoins. This is achieved by placing the receipts for the coins of a given node in the message that he has to forward. Thus, he cannot obtain the receipt unless his successor on the path decrypts them from the message and sends them to him.

Sender Pre-calculation Procedure

If S is a guest node, it can only use A-mcoins to pay the nodes in the path. In fact, S uses some of the A-mcoins (AC_k, r_k) it has withdrawn from the Bank to pay the first node in the path. Let $AC_{T_{i_\ell}}$ be the group of A-mcoins chosen by S for the specific communication round, namely $AC_{T_{i_\ell}} = \{AC_k : (AC_k, r_k) chosen\}$. Additionally, let $R^A_{T_{i_\ell}}$ be the set of the corresponding receipts (r_k-s). For all the nodes to be paid, it should hold that $|R^A_{T_{i_\ell}}| = \ell$, that is, the order of the receipt group equals the number of mcoins paid by S. If S is a Tor member, it can alternatively generate a group of random numbers $R_{T_{i_\ell}}$ of order ℓ, to create S-mcoins for paying node T_{i_ℓ}.

S, generates groups of random numbers — one for each node in the path — $R_{T_{i_{\ell-1}}}$, ..., $R_{T_{i_1}}$, where $|R_{i_j}| = j \; \forall j \in \{1, \ldots, \ell-1\}$. These groups correspond to payment-receipts of each node in the path. Moreover, these numbers will be enforced by S — through the secret communication channel S shares with all path nodes — to be used as S-mcoin-receipts for all S-mcoins spent throughout the payment round. Namely, R_{i_k} is of order k and its elements will serve as receipts for the mcoins T_{i_k} receives from its predecessor $T_{i_{k+1}}$.

For all the S-mcoin receipt groups R_{T_j}, where $j \in [1, \ldots, \ell-1]$, S calculates their H_r hashes

$$H_r(R_{i_{\ell-1}}), H_r(R_{T_{\ell-2}}), \ldots, H_r(R_{i_1})$$

where the same notation concerning $H_r(R_i)$s is used as before. If S uses S-mcoins, $R_{T_{i_\ell}}$ is also included in that set. By $H_r(R_X)$, where $X \in \{T_{i_\ell}, \ldots, T_{i_1}\}$, we denote the set $\{H_r(r_i)$ where $r_i \in R_X\}$. If S is a Tor node paying with S-mcoins, it additionally calculates $H_r(R_{i_\ell}$.

If S is part of the Tor network and wishes to use S-mcoins, it additionally calculates

$$SC_{S \to T_{i_l}}(r) = \{sig_S(MC, H_r(r), T_{i_l}) \forall r \in R_{T_{i_l}}\}.$$

Message Exchange Procedure

The general form of the message that a node $T_{i_{k+1}}$ sends a node T_{i_k} where $1 \leq k < l$ when forwarding the packets going on the path is as follows

$$\{T_{i_k}, \text{coins for } T_{i_k}, H(\text{coins for } T_{i_k}), \{M_{S \to T_{i_k}}\}_{K_{ST_{i_k}}}\}_{K_{T_{i_{k+1}} T_{i_k}}}.$$

The security of this message is guaranteed by the fact that it is signed with the common key $K_{T_{i_{k+1}} T_{i_k}}$ between $T_{i_{k+1}}$ and T_{i_k}. It specifies that the receiver of the message is T_{i_k} as well as the payment for him. The coins that are sent to T_{i_k} in this message are of the form $AC_{T_{i_l}}$ if they are A-mcoins or $SC_{T_{i_{k+1}} \to T_{i_k}}$ if they are S-mcoins. The corresponding hashes that verify the integrity if the coins used for payment here are $sig_S\{H(AC_{T_{i_l}}))\}$ and $sig_{T_{i_{k+1}}}\{H(SC_{T_{i_{k+1}} \to T_{i_k}})\}$ for the two types of coins respectively. The receiver is confident that the hashes are the right one because they are signed by the corresponding sender in each case.

We now describe the contents of the message $\{M_{S \to T_{i_k}}\}_{K_{ST_{i_k}}}$. As is obvious from the notation, this message carries information that is coming from the sender S and is meant to be read by the node T_{i_k}. The form of $M_{S \to T_{i_k}}$ is the following:

$\{T_{i_{k-1}},$
receipt for predecessor,
guarantee forT_{i_k}receipt,
value to generate payment for successor,
$\{M_{S \to T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}\}$

The above message tells T_{i_k} who is his successor $T_{i_{k-1}}$ on the forwarding path. Also it has the receipt that T_{i_k} has to send to his predecessor of the path so that $T_{i_{k+1}}$ can deposit his payment. This receipt has the form:

$$\{H_{ST_{i_{k+1}}}(R_{T_{i_{k+1}}}), R_{T_{i_{k+1}}}\}_{K_{ST_{i_{k+1}}}}.$$

Only the node $T_{i_{k+1}}$ can open the receipt since it is protected by the secret key it shares with sender; this also guarantees that it is coming from the sender. For integrity verification we have both the receipt $R_{T_{i_{k+1}}}$ and its hash. In the case when $T_{i_{k+1}}$ was paid with anonymous coins, $R_{T_{i_{k+1}}}$ is replaced by $R^A_{T_{i_{k+1}}}$

The next path in the message $M_{S\to T_{i_k}}$ is an integrity assertion that the coins that T_{i_k} has received from $T_{i_{k+1}}$ will have a corresponding receipt. This comes as

$$H_{ST_k}(H_r(R_{T_{i_k}}))$$

It is a hash value of the receipt that corresponds to the payment of T_{i_k}, protected by the key shared between T_{i_k} and S.

The message $M_{S\to T_{i_k}}$ further contains the hashed value $H_r(R_{T_{i_{k-1}}})$ that T_{i_k} will use to generate its payment to $T_{i_{k-1}}$. The last part in $M_{S\to T_{i_k}}$ is the corresponding message $M_{S\to T_{i_{k-1}}}$ from S to the successor $T_{i_{k-1}}$.

In the case when $T_k = R$ the message $M_{S\to T_{i_{k-1}}}$ is replaced by the actual message M sent from S to R. Additionally, no receipts are provided to R. How the last node in the path gets paid is discussed in the Discussion section.

If S uses A-mcoins, it sends the following message to T_{i_l} :

$$T_{i_l}, AC_{T_{i_l}}, sig_S\{H(AC_{T_{i_l}})\}, \{M_{S\to T_{i_l}}\}_{K_{ST_{i_l}}}$$

In particular, S sends to T_{i_l} the following:

$AC_{T_{i_l}}$: that is the A-mcoins to be spent.
$sig_S\{H(AC_{T_{i_l}})\}$: a signed hash H of the previous message to authenticate S and guarantee message integrity.
This part of the message can also be used for proving S's guilt in case it double-spends an A-mcoin.
$\{M_{S\to T_{i_\ell}}\}_{K_{ST_{i_\ell}}}$: a message to T_{i_ℓ} through the secret channel between T_{i_l} and S. The message has the following form :

$$T_{i_{\ell-1}}, H_{ST_{i_l}}(H_r(R_{T_{i_l}})),$$
$$H_r(R_{T_{i_{\ell-1}}}), \{M_{S\to T_{i_{\ell-1}}}\}_{K_{ST_{i_{\ell-1}}}}.$$

The exact meaning of each part of this message will be explained shortly.

A similar message is sent to T_{i_ℓ}, when S uses S-mcoins:

$$\{T_{i_\ell}, SC_{S\to T_{i_\ell}}(R_{i_\ell}), sig_S\{H(SC_{S\to T_{i_\ell}}(R_{i_\ell})),$$
$$\{M_{S\to T_{i_\ell}}\}_{K_{S'T_{i_\ell}}}\}_{K_{ST_{i_\ell}}}.$$

The general idea is that S will attempt to trick T_{i_ℓ}. In particular, S will try to pretend to T_{i_ℓ} to be an internal node in the path and that the actual initiator of the connection is a node S'. Under that umbrella, it creates two keys for T_{i_ℓ} regarding the same connection: $K_S T_{i_\ell}$ which S shares with T_{i_ℓ} as adjacent nodes in the specific connection and $K_{S'T_{i_\ell}}$ which the supposed initiator S' shares with T_{i_ℓ}. S sends to T_{i_ℓ} the following message encrypted with the secret key $K_{S'T_{i_\ell}}$.

$SC_{S\to T_{i_\ell}}$: that is the S-mcoins to be spent.
$H(SC_{S\to T_{i_\ell}})$: a hash H of the previous message to authenticate S and guarantee message integrity. This part of the message serves accountability reasons as well.
$\{M_{S\to T_{i_\ell}}\}_{K_{ST_{i_\ell}}}$: a message to T_{i_ℓ} - encrypted with the common secret key $K_{ST_{i_\ell}}$.

It is interesting to look into the structure of $M_{S\to T_{ell}}$ as it contains the following:

$T_{i_{l-1}}$: the next node in the path,

$\{H_{S'S}(R_S), R_S\}_{K_{S'S}}$: the receipt that corresponds to supposed coins given to S by the node previous to S in the path. This receipt is obviously fake and it is only part of message to T_{i_ℓ} if S has used S-mcoins in order to trick T_{i_ℓ}.

$sig_S\{H_{ST_{i_\ell}}(H_r(R_{T_{i_\ell}}))\}$: an integrity assertion that the hashed values — that basically the receipt related part of the S-mcoins — provided to T_{i_ℓ} by S through S's S-mcoins are the ones that is enforced by the initiator of the communication. In this way, T_{i_ℓ} can check nodes' honesty.

$H_r(R_{T_{i_{\ell-1}}})$: the hashed values to be used by T_{i_ℓ} when sending S-mcoins to $T_{i_{\ell-1}}$.

$\{M_{S\rightarrow T_{i_{l-1}}}\}_{K_{ST_{i_{l-1}}}}$: a message to the next node in the path encrypted with the key the initiator and $T_{i_{l-1}}$ node share.

In the general case, where node $T_{i_{k+1}}$ pays his successor T_{i_k}, it sends to the last, the following:

$$\{T_{i_k}, SC_{T_{i_{k+1}}\rightarrow T_{i_k}}(R_{Ti_k}),$$

$$sig_{T_{i_{k+1}}}(H(SC_{T_{i_{k+1}}\rightarrow T_{i_k}}(R_{Ti_k}))), \{M_{S\rightarrow T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}\}_{K_{T_{i_k}T_{i_{k-1}}}}$$

In particular, $T_{i_{k+1}}$ sends the following information to T_k, encrypted with the secret key $K_{T_{i_{k+1}}T_{i_k}}$ they share:

$SC_{T_{i_{k+1}}\rightarrow T_{i_k}}(R_{T_{i_k}})$: that is the S-mcoins to be spent. For simplicity purposes, we will denote this quantity as $SC_{T_{i_{k+1}}\rightarrow T_{i_k}}$ and we will assume that they were created using the Recipient's receipt set $R_{T_{i_k}}$.

$H(SC_{T_{i_{k+1}}\rightarrow T_{i_k}}(R_{T_{i_k}}))$: a hash H of the previous message to authenticate S and guarantee message integrity. This part of the message serves accountability reasons as well.

$\{M_{S\rightarrow T_{i_k}}\}_{K_{ST_{i_k}}}$: a message to T_{i_k} - encrypted with the common secret key $K_{ST_{i_k}}$.

S, provides to node T_{i_k} of the path — secretly through $M_{S\rightarrow T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}$ —, the following information:

$T_{i_{k-1}}$: the next node in the path,

$\{H_{ST_{i_{k+1}}}(R_{T_{i_{k+1}}}), R_{T_{i_{k+1}}}\}_{K_{ST_{i_{k+1}}}}$: that is the the receipt for the mcoins spent from T_{k+2} to $T_{i_{k+1}}$. We will denote this receipt packet as $Rcp(T_{i_{k+1}})$. As we can see, $Rcp(T_{i_{k+1}})$ is encrypted with the secret key S and T_{k+1} share. Also, it contains another hash of the receipt — other than H, that was used for the verification part $VR_{T_{i_{k+1}}}$ of mcoins. This is done so that integrity can be assured.

$H_{ST_{i_k}}(H_r(R_{T_{i_k}}))$: an integrity assertion that the hashed values given to T_{i_k} by T_{k+1} in the receipt related VR part of $T_{i_{k+1}}$'s s-mcoins are the ones provided by S. T_{i_k} can then check T_{k+1} 's honesty.[2]

$H_r(R_{T_{i_{k-1}}})$: the hashed value to be used by T_{i_k} when it spends coins to $T_{i_{k-1}}$, which will from now be denoted as $VR_{T_{i_{k-1}}}$.

$\{M_{S\rightarrow T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}$: a message to the next node in the path.

T_{i_k}, upon receiving the message described above, it decrypts the total message using $K_{T_{i_{k+1}}T_{i_k}}$. It checks message integrity via the hashes and S-mcoin's validity by verifying the $T_{i_{k+1}}$'s signature on them as well as whether $RV_{T_{i_k}}$ is the proper one. Then T_{i_k} sends the following message to T_{k-1}:

$$\{T_{i_{k-1}}, SC_{T_{i_k}\rightarrow T_{i_{k-1}}}, [sig_{T_{i_k}}(H(SC_{T_{i_k}\rightarrow T_{i_{k-1}}})),$$

$$\{M_{S\rightarrow T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}\}_{K_{T_{i_k}T_{i_{k-1}}}},$$

where $M_{S\rightarrow T_{i_{k-1}}}$ = part has the form:

$$T_{i_{k-2}}, Rcp(T_{i_k}), H_{ST_{i_{k-1}}}(RV_{T_{i_{k-1}}})), RV_{T_{i_{k-2}}},$$

[2] Here, we implicitly assume that S is honest. If not, then there is a way for S in collaboration to a node in the path to trick an intermediate path node. We address this problem and how we handle it in discussion section.

$$\{M_{S\rightarrow T_{i_{k-2}}}\}_{K_{ST_{i_{k-2}}}}$$

While this procedure continues, we finally get to the last node in the path T_{i_1}. That node finally gets message M encrypted to the receiver's public key and submits the packet to R. Namely, message that T_{i_1} from his predecessor T_{i_2} receives is:

$$\{T_{i_1}, SC_{T_{i_2}\rightarrow T_{i_1}}(R_{Ti_1}),$$
$$sig_{T_{i_2}}(H(SC_{T_{i_2}\rightarrow T_{i_1}}(R_{Ti_1}))), \{M_{S\rightarrow T_{i_1}}\}_{K_{ST_{i_1}}}\}_{K_{T_{i_2}T_{i_1}}}$$

where

$$\{M_{S\rightarrow T_{i_1}}\}_{K_{ST_{i_1}}} = \{R, \{H_{ST_{i_2}}(R_{T_{i_2}}), R_{T_{i_2}}\}_{K_{ST_{i_2}}}\{M\}_{pk_R}.$$

and after the appropriate validation procedures, T_{i_1} sends a receipt to T_{i_2}:

$$\{\{H_{ST_{i_2}}(R_{T_{i_2}}), R_{T_{i_2}}\}_{K_{ST_{i_2}}}\}_{K_{T_{i_2}T_{i_1}}}.$$

Deposit At each deposit time the nodes deposit all mcoins that they have received during the period. Detailed analysis of the deposit rate will be given in the next section. Here we define the technical procedure to be followed given a fraction τ of total mcoins that are deposited for a given node T_i.

- S-coins will be deposited in one of the standard ways for deposit of microcoins; double-spending will be prevented as with microcoins [8]

- A-mcoins: All A-mcoins are depositable for their nominal value.

3.3 Discussion

There are some issues that we need to pay attention to.

- Receipt-packets have the form of
$$\{H_{ST_{i_k}}(R_{T_{i_k}}), R_{T_{i_k}}\}_{K_{ST_{i_k}}}$$
That is, they contain the random numbers that constitute the actual receipt, as well as a hash of them that has been agreed between node T_{i_k} and S, $H_{ST_{i_k}}$. This hashed value, is used for integrity purposes.
- Mcoins are in any case hashed with the global hash H and then signed by their sender. This is done for two reasons, integrity and accountability. If double-spending has occurred and is related to that mcoin, the signature on the hash proves who double-spent it. For efficiency reasons, we make the sender sign the hash instead of signing the whole group of coins. Except for the messages exchanged, however, there are many issues we need to discuss and deal with. From now on, and without loss of generality, we will assume that all nodes up to node T_{i_k} in the path have behaved honestly, namely they forwarded the payment packet the way they were supposed to.
- One issue to think about is how do we make T_{i_k} forward the packet to the next node in the path instead of sending them elsewhere. The answer is that this is the only way for T_{i_k} to get paid. Since nodes $\ell, \ldots, k+1$ are assumed to be honest, T_{i_k} has received from node $T_{i_{k+1}}$ k S-mcoins. The receipts that correspond to these S-mcoins are provided by S to node $T_{i_{k-1}}$ through the secret channel that nodes $T_{i_{k-1}}$ and S have created ($\{M_{S\rightarrow T_{i_{k-1}}}\}_{K_{ST_{i_{k-1}}}}$). Thus, in order for T_{i_k} to be able to receive the receipt for the S-mcoins it has received from its predecessor, it has to forward the message to its successor $T_{i_{k-1}}$.
- In the preceding discussion, we assumed that S is honest. What if S decides to play unfairly? Assume, for example, that S still wants to send his traffic, but tries to pay less than it should. Suppose that S colludes with T_{i_k}. T_{i_k} sends to $T_{i_{k-1}}$ S-mcoins whose receipt validation parts match the hashed values sent to $T_{i_{k-1}}$ by S. Suppose now that S has provided wrong receipts to $T_{i_{k-2}}$ and thus $T_{i_{k-2}}$ remains unpaid, although he has given his mcoins to $T_{i_{k-2}}$. In this case, $T_{i_{k-2}}$ pays for S's traffic. In order to prevent this attack from happening, we adopt a prepayment scheme. According to this property, path-nodes are prepaid during the connection establishment procedure for a fixed number of packets. In this case, the previous attack does not work, since $T_{i_{k-2}}$ would tear the connection down after finding out the trick and S not be able to send its packets.

– There is also an issue with the last node in the path. He is the only node to be paid in the next packet round. We resolve that by placing in the first data packet the receipts for the mcoins it has received.
– We make the assumption that nodes in general want to make profit. Thus it is of no interest of them to spoil the connection.
– Double-spending of anonymous coins is caught at deposit time. The amount of double-spending is controlled by the deposit rate, which will be discussed in detail in the next section.

4 System Security

A system's security depends heavily on the capabilities of its adversaries. Therefore, for each of the cases described in this section, we clearly describe both the threat model, the compromised entities, and the severity of the attack. There has been a wealth of research related to attacks against onion routing systems including Tor. Our effort is to validate that OpenTor does not introduce new types of attacks, especially ones that can target either the anonymity or the robustness of an onion routing system. We start with attacks that attempt to expose the sender's anonymity.

4.1 Correctness-Robustness

This property requires that if all Users behave legally, all the packets will reach their final destination and all nodes in the path will be paid for the service they offered. This is assured by two facts:

– Consider the case where all nodes are honest, all nodes have been prepaid, and all forward the packet traffic to the next hop in the path. Int that case, packets do reach their destination.
– Each node pays its successor one coin less than the coins it receives from its predecessor. Legal nodes produce legal microcoins and use anonymous coins only once. Thus, payment is executed correctly.

Correctness: *Proof.* The proof follows directly from *Lemmas 1-4.*

Lemma 1. If all primitives regarding micropayments hold and nodes are honest, then S-mcoins and A-mcoins that they generate are valid.

Lemma 2. If all the underlying primitives for micropayments and blind signatures hold, the payment protocol is executed correctly, and all participants in a path are honest, then all nodes in that path will be paid an mcoin for each group of packets to be forwarded.

Lemma 3. If all nodes in the path are honest and have paid as they should, the packets will reach their destination.

Lemma 4. If all the underlying primitives for blind signatures hold, the anonymity of the sender is cryptographically preserved.

Robustness: The proof of robustness in our system is given in Appendix. However, we state here some interesting results concerning that property. Let α be the fraction of malicious nodes participating in Tor. By "malicious" we mean nodes that would intentionally cause the connection to stop. Per the calculations contained in the Appendix, we conclude that the probability that a User succeeds in sending his traffic — namely no malicious node exists in the path the User chooses — is the following:

$$P(x \notin P_\ell \forall x \in X) = (1-\alpha)^\ell.$$

where we make the assumption that the number of nodes in Tor that particular moment $n \gg 1$. $X = \{x_1, \ldots, x_{\alpha n}\}$ is the set of malicious nodes, P_ℓ any path of ℓ Tor nodes.

The plots for the values $\ell = 3, \ell = 4, \ell = 5$ and $\ell = 6$ are given in Figure 2. For each one of these diagrams, we assume $n \simeq 1000$, and observe the change in probability that the path is clean while the fraction of malicious users in Tor changes. For example, if $\alpha = .1$, the probability of a secure path is .73 if $\ell = 3$ but only .53 if $\ell = 6$. Increasing the path length has cut the probability of a secure path by about 27%. Another issue is that by making Tor "open" and hence letting anyone participate, we increase the chance of

having malicious nodes in the system. The most dangerous case is when the first node in a path is malicious; it can reveal the sender and in collusion with receiver can expose the connection between sender and receiver. Taking this into consideration, we quantify the anonymity of the system by the probability that the first node in a path is not malicious: $1 - \alpha$.

4.2 Monetary Unit Unforgeability

The Unforgeability property guarantees that no fake coins can be deposited in the system. We have two types of coins:

- **S-mcoin Unforgeability** : For node T_i to fore an S-mcoin $SC_{T_j \to T_i}$, it would have to forge the signature of T_j. It cannot do this.
- **A-mcoin Unforgeability**: Node T_i cannot forge an A-mcoin $AC(r_i)$ since it cannot forge Bank's signature.

Coin Unforgeability is achieved in our scheme. *Proof.* I t derives from the following lemmas.

Lemma5 If S-Mcoin Unforgeability and A-mcoin Unforgeability holds, then Coin Unforgeability is achieved.
*Proof.*It follows from the definition.

Lemma6 S-Mcoin Unforgeability and A-mcoin Unforgeability holds in our scheme.
*Proof.*It follows Lemmas 7 and 8.

Lemma7 If all the primitives for digital signatures hold, then "S-mcoin"- Unforgeability property is achieved.
Proof. See Appendix.

Lemma8 If all the primitives for blind signatures hold, then "A-mcoin"-Unforgeability property is achieved.
Proof. See Appendix.

4.3 Accountability

Accountability means that the identity of a node that behaves maliciously — double-spending, forging attempts, message manipulation, etc. — will be revealed, along with a proof of his guilt.

Accountability is achieved in our Scheme.

Proof. We only present a sketch of proof. It is achieved by the combination of two things :

- All nodes "know"[3] their predecessor and their successor nodes in a path.
- Each node participating in a path sends to its path-successor a signed hash of the message it sends. This is done for both for integrity and accountability reasons.

The above two facts allow each node to verify the signature of its predecessor; it can also use the message as proof of guilt if necessary.

The only case where accountability is limited, is when the initiator of a connection is malicious. If this is the case, although they cannot identify it, intermediate nodes do have proof of his guilt and can stop the connection.

4.4 Exculpability

Exculpability requires that an honest person cannot be framed by another user. Guilt in our system is always proven by digital signatures; therefore, it satisfies the exculpability requirement.

Proof. It derives from *Lemma9*.

Lemma9 If Accountability and primitives of digital/blind signatures hold, then Exculpability holds.

[3] By the word "know" we mean that every node in a path knows the public verification key of its path neighbors, which effectively identifies them.

4.5 Sender-Receiver Unlinkability and Path Confidentiality

The most serious attacks are the ones against the anonymity properties of the system. Such attacks attempt to reveal the network communications of the senders. We argue that in an onion routing system with payments, the Sender-Receiver Unlinkability requirement is preserved. In our analysis, we are not going to address the current weaknesses of the underlying anonymizing network. Rather, we study new attacks that stem from the addition of the payment scheme to existing onion routing systems including Tor. Our aim is to provide a formal understanding of potential exploitations of the payment scheme or the entities involved in the payment transactions when combined with known attacks against anonymity networks.

The new element in OpenTor is the existence of payments. The coins have to be withdrawn and deposited, potentially leaking information to an adversary. In OpenTor, the "honest but curious" Bank can play the role of such an adversary. Initially, we consider attacks against the anonymity of the communications that pass through the anonymity network, linking the senders to receivers. It is easy to show that mere knowledge of the payment transactions, even if they are timestamped, is not sufficient to link the senders to the receivers. Indeed, in our system the senders withdraw and pay with anonymous coins; thus, their identity cannot be revealed to the Bank. Furthermore, the receiver identities are not exposed since there is no monetary transaction between the last node in the path and the receivers. Therefore, to perform an attack, an adversary requires the collaboration of a Tor node and/or the receiver.

Another attack scenario is when there is collusion between the initial node T_L (L is the maximum path) and a compromised receiver R. Assuming that there is no dummy traffic that is terminated inside the OpenTor network, the two colluding nodes will be able to expose the identity of the sender. Indeed, by comparing the number of packets both T_L and R receive from the sender and the fact that the IP address of the sender is already known to the T_L, they can reveal the sender's anonymity. However, this attack does not depend on the operations or information leaked from the payment scheme but it is rather an attack against a generic onion routing network.

We now consider the following question: given an arbitrary node T participating in a connection $c^{S\rightarrow R}_{<T_{i_l},\ldots,T_{i_1}>}$, can we determine the node $\tilde{T}$ such that $T = T_i$ and $\tilde{T} = T_{i-1}$ for some $1 < i \leq l$? Since signed coins are cumulatively deposited, the bank can observe that the node T deposits the following number of coins coming from any of his neighbors T'

$$G(T', T) = \sum_{\forall c^{S\rightarrow R}_{<T_{i_{l'}},\ldots,T_{i_1}>} \in C(T',T)} k * c^{S\rightarrow R}_{<T_{i_{l'}},\ldots,T_{i_1}>},$$

where

$$C(T', T) = \{c^{S\rightarrow R}_{<T_{i_l},\ldots,T_{i_1}>} | \exists k \in \{1, \ldots l-1\}\ T' = T_{i_{k+1}},\ T = T_{i_k}\}$$

.

Also, the bank observes the number of coins signed by node T deposited at each of its successors T'' in various connections, which is :

$$G(T, T'') = \sum_{\forall c^{S\rightarrow R}_{<T_{i_{l''}},\ldots,T_{i_1}>} \in C(T,T'')} k * c_{<T_{i_{l''}},\ldots,T_{i_1}>},$$

The total number of packets forwarded by the node i is given by :

$$\sum_{T'} G(T', T) + G_{ac}(T) - \sum_{T''} G(T, T''),$$

where $G_{ac}(T)$ is the number of anonymous coins that T was paid as a first node in connections. However, $G_{ac}(T)$ is not known since the anonymous coins that T deposits include also his own coins; T deposits these in order to preserve anonymity. Thus, we can determine neither the number of packets forwarded through the node nor the position of the node in the connections that it participates into.

Let us now assume that some predecessor T' of the node T in some of the connections it participates in is malicious and reveals to the bank all the information about its incoming and outgoing connections. Thus,

the bank will know $G(T', T)$. The only extra information that bank can infer based on this knowledge is that packets $c_{<T_{i_{l'}},\ldots,T_{i_1}>^{S\to R}}$ where $T = T_k$ cannot be forwarded to a node $\tilde{T}$ if:

$$(k-1) * c^{S\to R}_{<T_{i_{l'}},\ldots,T_{i_1}>} > G(T, \tilde{T}),$$

i.e., the node $\tilde{T}$ received fewer coins from T than the necessary amount to pay for the forwarding of the packets along the connection. To evaluate the importance of the above restriction on the possible forwarding nodes after T, we consider the average case for all the rest of the connections passing through node T.

4.6 Deposit Rate

In the previous section, we considered the question of how an adversary having a global view of the deposits made in the OperTor network can exploit this information for tracing packet paths. An implicit assumption is that the Bank receives the transaction information in certain time intervals. These intervals reflect the activity of the network for a certain period of time, i.e. the accumulation period determined by the deposit rate of the signed coins in the system. Of course, this time periods are dependent on the coin deposit rate and are of essential importance to thwart attacks that try to exploit payment transaction information. Here, we analyze what should the deposit rate be to minimize or even eliminate attacks that attempt to break anonymity based on the payment transactions received by the Bank.

We start with the assumptions that the number of packets on the paths in the network are drawn from a uniform distribution, which implies that the aggregated numbers of coins at each node and on each edge will follow a normal distribution. Let μ and σ be the mean and standard deviation for the normal distribution of packet payments going through a node and $\tilde{\mu}$ and $\tilde{\sigma}$ be the mean and standard deviation for the normal distribution of packet payments going through an edge. Since the maximum path length of a connection in the OpenTor network is L and for a particular connection $c^{S\to R}_{<T_{i_l},\ldots,T_{i_1}>}$ a node T has probability $\frac{1}{n}$ to be the first, second, $\ldots, l-th$ node in the path, therefore

$$\mu = (\frac{1}{n} + \frac{2}{n} + \cdots + \frac{L}{n}) * \sum_{\forall c^{S\to R}_{<T_{i_l},\ldots,T_{i_1}>} \in C(T)} c^{S\to R}_{<T_{i_l},\ldots,T_{i_1}>} =$$

$$\frac{L*(L+1)}{2n} * \sum_{\forall c^{S\to R}_{<T_{i_l},\ldots,T_{i_1}>} \in C(T)} c^{S\to R}_{<T_{i_l},\ldots,T_{i_1}>}.$$

Since we are looking at the average case when we have a total of N packets sent in the network on total of C connections, we have

$$\mu = \frac{N*L*(L+1)}{2n^2}.$$

The worst case scenario that the coins come into a node on only $n-1$ edges, then we must have:

$$\tilde{\mu} = \frac{N*L*(L+1)}{2n^2*(n-1)}.$$

To proceed, we need to identify the accumulated amount of coins at a node so that maximal size of the packets transfered on a single incoming connection into the node is smaller than the minimum total of the number of packets transfered on each of the outgoing connections from that node. We would like to account for all the connections that within $k*\sigma$ range of the expected value. Thus the maximum number of packets coming into a node T is $\mu + k*\sigma$. On the other hand the minimum aggregated number of packets on each of the outgoing edges will be $\tilde{\mu} - k*\tilde{\sigma}$, or $(n-1)*(\tilde{\mu} - k*\tilde{\sigma})$ for all of the outgoing connections. So now we have the following requirement:

$$\tilde{\mu} - k*\tilde{\sigma} > \frac{N}{C} * \frac{L*(L+1)}{2}$$

If we account for the fact that the payment coming on a single connection does not exceed the aggregated payment on any of the outgoing edges. This implies that the total number of connections C established in

the network should be of the order $O(n^3)$ and each node should participate on average in $O(n^2)$ of them. Thus a node should deposit signed coins when it has participated in this number of connections. Here we want to point out that the so derived conditions provide that for a certain connection we cannot rule out any possible path, which is a much stronger result than not to be able to disclose the exact route of the packets. For a connection of length L there are n^L possible paths and since we are assuming a malicious node in the path we have at least n^{L-1} possible paths. Thus $O(n^2)$ connections going through a node T guarantee that no other node can be ruled out as its possible successors on a particular connection.

Now we will consider the case when we allow to be able to eliminate some the nodes as possible successor nodes in the path of a connection but still want to keep some constant number n_c of possible successors. In this case we restrict our attention only to those edges assuming that the coins coming on the rest of the edges are negligible and in this case we have:

$$\tilde{\mu} = \frac{N * L * (L+1)}{2n * n_c * (n-1)}.$$

Now the condition that the payments on each of this edges exceeds the payment for the packets a particular connection translates into

$$\frac{N * L * (L+1)}{2n * n_c * (n-1)} - k * \tilde{\sigma} > \frac{N}{C} * \frac{L * (L+1)}{2}$$

implying $O(n^2)$ connections in the network in total or $O(n)$ connections through a node. And we still have n_c^{L-1} possible paths for a particular connection. The number n_c will depend on the size of the incoming packet flow that we consider as well as on the average number of connections going through an edge. The connections using a given edge $\frac{2C}{n(n-1)}$. Thus in the case when the packet flows are within two standard deviations of the expected value, then we can assume that any outgoing edge with payment on at least $\tilde{\mu} + \frac{n(n-1)}{2C} * \tilde{\sigma}$ cannot be eliminated as a possible outgoing edge for the flow. When $C = n^2$ we expect the edges with payment exceeding $\tilde{\mu} + \tilde{\sigma}$, which implies $n_c = 0.16 * n$ for normal distribution.

At each time of deposit in the system a node deposits both signed and anonymous coins. Since anonymous coins come as payments for the first nodes in the connection paths and signed coins are used for the payments for all the rest of the nodes in the connection paths, the ratio between anonymous coins that were used for actual payment (excluding the own anonymous coins that are used to preserve anonymity) and the signed coins used in the system should be $\frac{2}{L-1}$ since for each packet we use L anonymous and $\frac{2L}{L-1}$ signed coins. Our goal is to fix the deposit frequency in such a way that the ratio of anonymous vs. signed coins deposited by each node is also $\frac{2}{L-1}$. However, since we want to account for the case when we have connections with number of packets in range of two standard deviation of the mean value for the distribution, we know that we will have nodes which will have ratio of anonymous to signed coins between $\frac{2L-k*\sigma}{L-1}$ and $\frac{2L+k*\sigma}{L-1}$. In the case when we have less anonymous coins the node can always use its own anonymous coins to pad, we require that the actual deposit ratio of anonymous vs. signed coins be $\frac{2L+k*\sigma}{L-1}$. Thus the expected total amount of coins for each deposit of a node will be $\frac{2L+k*\sigma}{L-1} * \frac{N}{C}$ where have determined the number of connections established during one deposit period according to the restrictions we have derived above and N is the total number of packets expected to be transfered in the network.

5 System Evaluation

In this section we attempt a rough estimate of our system's performance. We do this in two ways. First, we show the number of extra packets that need to be sent for the payment procedure to take place. Next, we consider the number of hash, digital signature and encryption operations that need to be computed for the transmission of N packets, which can be paid with one coin at each node. We call this set of payment operations a "payment round". For the transmission of N packets by each node, the following payment messages exchanged:

- l packets are transmitted containing the mcoins from one node to its successor in the path (one packet for each edge).

– l transmissions for the receipts of the l mcoins

Based on this, the number of extra packets transmitted for the payment of utilizing Tor Anonymous service is $2l$ and the percentage of the extra traffic is $\frac{2l}{2N} = \frac{l}{N}$.

Thus, the smaller the value of an mcoin is, in terms of packets, the bigger the system extra load would be.

We now present a sketch of the system's performance evaluation in which we assume that

– c_h is the cost of a hash function.
– c_e is the cost of a symmetric encryption procedure.
– c_s is the signature cost.

We consider a path $\langle T_{i_l}, T_{i_{l-1}}, \ldots, T_{i_1} \rangle$ of length l. The cost of producing a signed coin, including the receipt, is:

$$c_{SC} = c_s + c_s = 2c_s.$$

The corresponding cost for an anonymous coin c_{AC} is then

$$c_{AC} = c_s.$$

For every group of N packets to be sent and for each node T_{i_k} in the path, S calculates:

– hashes of k random numbers, which cost kc_h
– k verifications of the hashes given as payment to each node: $kc_h h$
– hashes of k receipts, which are encrypted with the receipt into a receipt packet. This procedure costs $c_e + kc_h$.
– the encryption of the secret message it sends to all nodes in path, whose cost is simply c_e.

Consequently, the total cost of S's pre-calculation for transmitting a packet is :

$$\begin{aligned} c_{pr} &= \sum_{k=1}^{k=l} \{(kc_h + kc_{hh} + kc_h + c_e + c_e)\} \\ &= \frac{k(k+1)}{2}(2c_h + c_{hh}) + 2kc_e \end{aligned}$$

Additionally, the message creation from one node T_{i_k} to the next hop $T_{i_{k-1}}$ is

$$\begin{aligned} c_{k \to k-1} &= (k-1)c_{SC} + (k-1)(c_s + c_h) + c_e \\ &= (k-1)(3c_s + c_h) + c_3 \end{aligned}$$

where

$(k-1)c_{SC}$ is the cost for creating $(k-1)$ signed microcoins,
$(k-1)(c_s + c_h)$ is the cost for hashing and signing the hash of each S-mcoin, and
c_e is the cost for encrypting the whole message.

Consequently, the total cost for the transmission of a set of N packets $C(N)$ is :

$$\begin{aligned} C_N &= \sum_{m=1}^{m=l} \{c_{k\to k-1}\} + c_{pr} \\ &= \sum_{m=1}^{m=l} \{(k-1)(3c_s + c_h) + c_e\} \\ &\quad + \frac{k(k+1)}{2}(2c_h + c_{hh}) + 2kc_e \\ &= \frac{(l-1)l}{2}(3c_s + c_h) + (l-1)c_e \\ &\quad + \frac{l(l+1)}{2}(2c_h + c_{hh}) + 2lc_e \\ &= \frac{3l(l-1)}{2}c_s + \frac{(3l+1)l}{2}c_h + \frac{l(l+1)}{2}c_h h + (3l-1)c_e. \end{aligned}$$

The total cost, then, is $\mathrm{O}(l^2 + l)$.

6 Conclusion

Current anonymity networks appear to lack wide participation due to their volunteer nature. We believe that by providing economic incentives, we can help incentivize users to both participate and to use anonymity networks to protect their communications. Unfortunately, current payment schemes cannot be used unmodified to enable payments in Tor.

To address that, we design a novel hybrid scheme and we prove that it is possible to add a secure payment scheme to Tor. The cost of the scheme is dominated by two factors, the path length and the number of packets per payment. However, the two have very different properties. The number of packets per payment, N, represents the tradeoff between performance and risk. By setting N high, the total cost of our scheme is minimized, since the expense is amortized over a large number of transmissions. However, N also represents how willing nodes are to transmit packets without assurance of payment. If N is too high, a cheater can send a fair amount of data before being caught. We expect that market forces will determine the proper value.

The path length, L, is a more interesting value. Looking at it purely in terms of our payment scheme, it represents the tradeoff between performance and anonymity. Buying more anonymity is indeed expensive in our scheme. However, as we have shown, increasing L too much *decreases* security if we assume that some percentage of nodes cannot be trusted. The maximum value of L is thus capped by that problem, thereby bounding the cost of the payment scheme.

References

1. Chaum, D., Fiat, A., and Naor, M. Untraceable electronic cash. In *Proceedings of CRYPTO '88* (1988).
2. Chaum, D. L. Untraceable Electronic Mail, Return Addresses, and Digital Psuedonyms. *Communications of the ACM* (1981).
3. Diffie, W., and Hellman, M. E. New directions in cryptography. *IEEE Transactions on Information Theory IT-22*, 6 (1976), 644–654.
4. Dingledine, R., Mathewson, N., and Syverson, P. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium* (August 2004).
5. Goldschlag, D. M., Reed, M. G., and Syverson, P. F. Hiding routing information. In *Workshop on Information Hiding* (May 1996), R. Anderson, Ed., Springer-Verlag, pp. 137–150. LLNCS 1174.
6. Jakobsson, M., and Juels, A. An optimally robust hybrid mix network (extended abstract). In *Proceedings of Principles of Distributed Computing - PODC '01* (2001), ACM Press.

7. Kesdogan, D., Agrawal, D., Pham, V., and Rautenbach, D. Fundamental limits on the anonymity provided by the mix technique. In *Proceedings of the IEEE Symposium on Security and Privacy* (2006).
8. Micali, S., and Rivest, R. L. Micropayments revisited. In *CT-RSA* (2002), pp. 149–163.
9. Möller, U., Cottrell, L., Palfrader, P., and Sassaman, L. Mixmaster Protocol — Version 2. Draft, July 2003.
10. Øverlier, L., and Syverson, P. Locating hidden servers. In *Proceedings of the IEEE Symposium on Security and Privacy* (2006).
11. Rivest, R. Peppercoin micropayments, 2004.

A Detailed Proofs

Proof of Lemma1.
Validity and Depositability of S-mcoins is strongly connected to microcoins, since validity check is done on a part of S-mcoin which is an actual micro-coin. On the other hand, validity of A-mcoin depends on the Bank that is considered to be an trusted entity in regards of A-mcoin generation. Since, all the nodes in our system are honest, mcoins created are valid.

Proof of Lemma2.
This comes directly from *Lemma1* and the fact that each node in the path spends exactly one mcoin less than the mcoins it receives from its predecessor.

Proof of Lemma3.
It derives directly from correctness of the underlying anonymizing system. If nodes are honest and get paid, they perform.

Proof of Lemma4.
A-mcoins are only provided by the initiator of a communication channel. Since, A-mcoins are basically blind signatures on random numbers, they reveal no information on the party who signed the message or the party who withdrew them. Thus when deposited to the Bank and since we assume full graph, Bank cannot trace the Sender. On the other hand, when S-mcoins are used by the initiator S, since S fakes receipts etc., there is no possible way of linking him to the initiator[4].

Proof of Monetary-Unit Unforgeability.
It derives from *Lemma5–7.*
Lemma5. *If S-Mcoin Unforgeability and A-mcoin Unforgeability hold, then Coin Unforgeability is achieved.*
*Proof.*It follows from the Coin Unforgeability definition.
Lemma6. *S-Mcoin Unforgeability and A-mcoin Unforgeability hold in our scheme.*
*Proof.*It follows *Lemmas7–8.*
Lemma7. *If all the primitives for digital signatures hold, then "S-mcoin"- Unforgeability property is achieved.*
Proof. Let that system node T_i has managed to forge an S-mcoin $SC_{j\to i} = sig_{T_j}\{MC, H_r(k), T_i\}$ of node T_j without having any information about secret key of T_j. This would mean that T_i is able to sign on behalf of T_j. However, this is computationally impossible for all known digital signature schemes.
Lemma8. *If all the primitives for blind signatures hold, then "A-mcoin"-Unforgeability property is achieved.*
Proof. Let that node T_i managed to forge an A-mcoin $AC(r_i)$ without knowing Bank's secret key of Bank. This implies that T_i is able to sign on behalf of Bank which contradicts digital signature scheme properties.

Proof of Exculpability
*Proof.*Directly from *Lemma9.*
Lemma9. *If Accountability and primitives of digital/blind signatures hold, then Exculpability holds.*
*Proof.*Accountability provides proof on someone behaving in a bad way. This proof is basically digital signatures. To frame a user, one should be able to forge these, which is computationally impossible.

[4] We will refer to this case in a later section

Proof of Robustness results.

Proof. Let α be the fraction of malicious nodes participating in Tor. With the term "malicious" we denote nodes that would cause on purpose the connection to stop. Therefore the probability of a node in a bad to behave in a non-malicious way is $1 - \alpha$. Therefore the probability that a path of length l does not contain a malicious node is $(1-\alpha)^l$. Now if L is the maximum path length in the system, the probability that no path contains malicious nodes becomes:

$$\sum_{l=3}^{L}(1-\alpha)^l = \frac{1}{L-3}\frac{(1-\alpha)^{L+1} - (1-\alpha)^3}{\alpha}$$

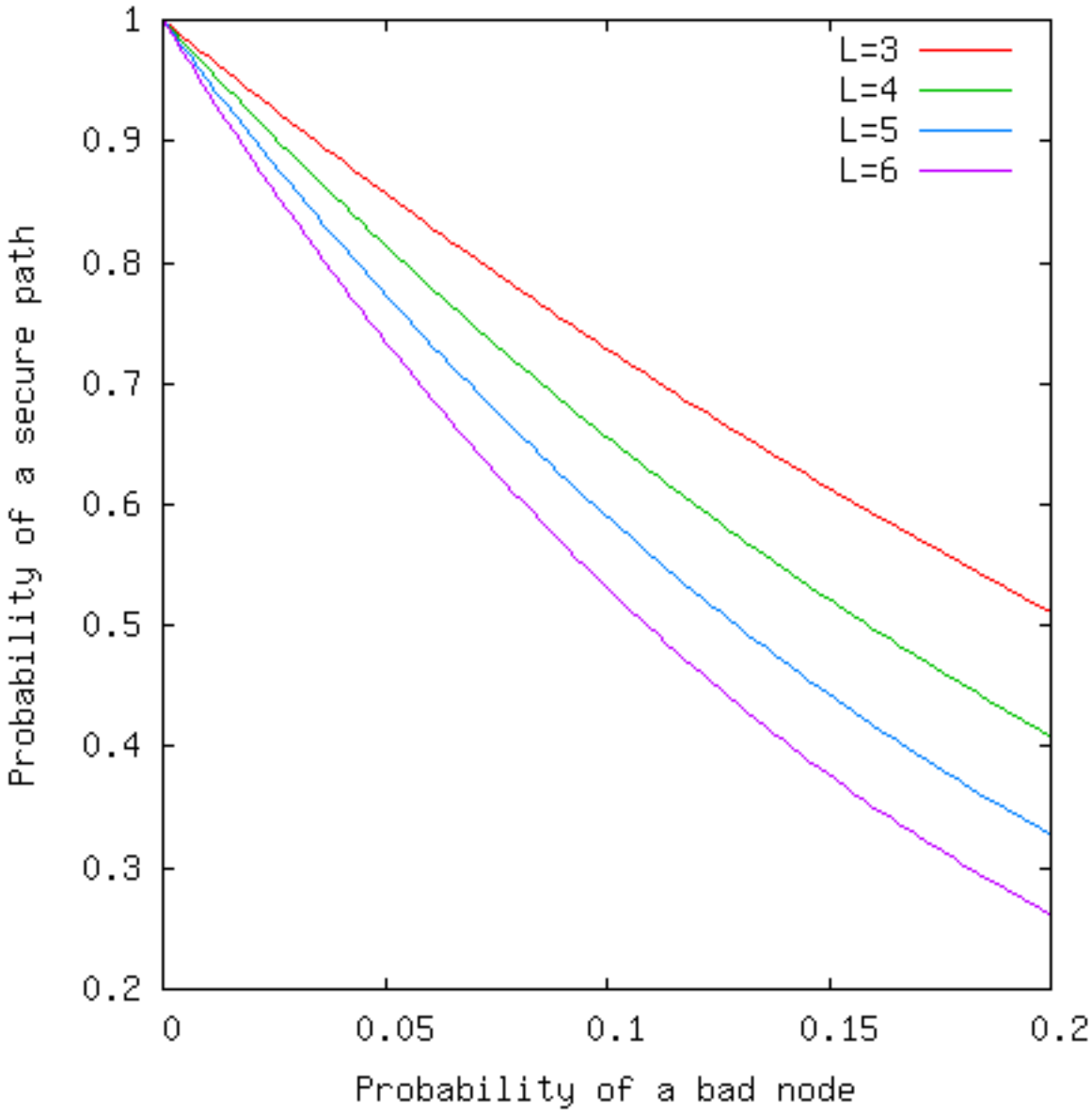

Fig. 2. The probability of a secure path, for varying path lengths and probabilities of a malicious user.