A Model of Onion Routing with Provable Anonymity

Joan Feigenbaum[1⋆], Aaron Johnson[1⋆⋆], and Paul Syverson[2⋆⋆⋆]

[1] Yale University {Joan.Feigenbaum, aaron.johnson}@yale.edu
[2] Naval Research Laboratory syverson@itd.nrl.navy.mil

Abstract. Onion routing is a scheme for anonymous communication that is designed for practical use. Until now, however, it has had no formal model and therefore no rigorous analysis of its anonymity guarantees. We give an IO-automata model of an onion-routing protocol and, under possibilistic definitions, characterize the situations in which anonymity and unlinkability are guaranteed.
Keywords: Security, privacy, anonymity, onion routing

1 Introduction

Anonymity networks allow users to communicate while hiding their identities from one another and from third parties. We would like to design such networks with strong anonymity guarantees but without incurring high communication overhead or much added latency. Many designs have been proposed that meet these goals to varying degrees [1].

Of the many design proposals, onion routing [8] has had notable success in practice. Several implementations have been made [8, 13, 6], and there was a similar commercial system, Freedom [2]. As of September 2006, the most recent iteration of the basic design, Tor [6], consists of over 750 routers, each processing an average of 100KB/s. Onion routing is a practical anonymity-network scheme with relatively low overhead and latency. It provides two-way, connection-based communication and does not require that the destination participate in the anonymity-network protocol. These features make it useful for anonymizing much of the communication that takes place over the Internet today, such as web browsing, chatting, and remote login.

Many Tor users communicate with web-based businesses and financial services. Chaum [4] was the first to note that even the best ecash design fails to be anonymous if the network identifies the customer. Even if a client is not hidden from the service, *e.g.*, she's using ordinary credit cards, she may desire privacy from her network-service provider, which might be her employer or just an ISP that is not careful with logs of its users' activities. Examples of the threat posed by both of these situations have been all too frequent in the news. Businesses

⋆ Supported in part by NSF grants 0331548 and 0428422.
⋆⋆ Supported by NSF grant 0428422.
⋆⋆⋆ Supported by ONR.

also make integral use of Tor to protect their commercial interests from competitors or to investigate the public offerings of their competitors without being observed. One vendor discovered by using Tor that its competitor had been offering a customized web site just for connections from the vendor's IP address.

Low latency and other performance characteristics of Tor can be demonstrated experimentally; anonymity-preserving properties cannot. Also, even with careful design, vulnerabilities can persist. The initial Tor authentication protocol had a cryptographic-assumption flaw that left it open to man-in-the-middle attacks. The revised authentication protocol was then proven to be free of such flaws [7]. As Tor is increasingly relied upon for sensitive personal and business transactions, it is increasingly important to assure its users that their anonymity will be preserved. Long-established components of such assurance in system security include a formal model, proving security guarantees in that model, and arguing that the model captures essential features of the deployed system. These are what we provide in this paper.

An onion-routing network consists of a set of onion routers and clients. To send data, a client chooses a sequence of routers, called a *circuit*, and constructs the circuit using the routers' public keys. During construction, a shared symmetric key is agreed upon with each router. Before sending data, these keys are used to encrypt each packet once for each router in the circuit and in the reverse of the order that the routers appear in the circuit. Each router uses its shared key to decrypt the data as it is sent down the circuit so it is fully decrypted at the end. Data flowing up to the client has a layer of encryption added by each onion router, all of which are removed by the client. The layered encryption helps hide the data contents and destination from all but the last router and the source from all but the first router. The multiple encryption and decryption also makes it harder for an observer to follow the path the data takes through the network.

Anonymity has not yet been rigorously proven of onion routing. We thus propose a formal model of onion routing based on the Tor protocol and analyze the anonymity it provides. Our model is expressed using IO automata [9], which provide us with asynchronous computation and communication. We then suggest definitions of anonymity and unlinkability with respect to an adversary within this model. The adversary is local and active in that he controls only a subset of the routers but can perform any arbitrary computation with them. This is the adversary against which Tor was designed to protect [6] and is a good adversary model for the situation in which Tor is currently running. Finally, we provide necessary and sufficient conditions for anonymity and unlinkability to be provided to a user in our model. It should be noted that we do not analyze the validity of the cryptography used in the protocol and instead base our proofs on some reasonable assumptions about the cryptosystem.

We only consider possibilistic anonymity here. An action by user u is considered to be anonymous when there exists some system in which u doesn't perform the action, and that system has an execution that is consistent with what the adversary sees. The actions for which we consider providing anonymity are sending messages, receiving messages, and communicating with a specific destination.

More refined definitions of anonymity, [12, 5], incorporate probability. We leave for future work applying such definitions to our system, which could be done by defining a probability measure over executions or initial states. Also, for simplicity, the model includes almost no concept of time. We do add circuit identifiers to mimic an attacker's ability to do timing attacks: the observation of distinctive timing patterns in a stream of data, whether inherent or attacker-induced. There is no timestamp included with actions, though, so there is only an ordering on actions.

The main result we show is that the adversary can determine a router in a given user's circuit if and only if it controls an adjacent router, with some other minor conditions. In particular, the adversary can determine which user owns a circuit only when the adversary controls the first hop. The set of users which have an uncompromised first hop form a sender "anonymity set," among which the adversary cannot distinguish. Similarly, the adversary can determine the last router of a circuit only when it controls it or the penultimate router. Such circuits provide receiver anonymity. Also, a user is "unlinkable" to his destination when he has receiver anonymity or his sender anonymity set includes another sender with a destination that is different or unknown to the adversary.

The first-hop/last-hop attack is well-known [13], but we state it in full detail and show that, in a reasonable formal model, it is the only attack that an adversary can mount. Also, our results persist with or without some of the nontrivial design choices, such as multiple encryption and stream ciphers. This doesn't imply that these features are unnecessary – they may make attacks more difficult in practice and meet other goals such as data integrity, but it does illuminate their effect on the security of the protocol. Finally, we present the first formal network model and protocol definition for onion routing, and give definitions for anonymity and unlinkability within that model.

2 Related Work

Numerous papers have informally discussed the security of the design of onion routing and related systems, as well as theoretical and experimentally demonstrated attacks. There have also been numerous formalizations of anonymous communication. However, formal analyses have primarily been of systems other than onion routing, *e.g.*, DC nets and Crowds. (Cf. [1] for examples of all of these.)

Recent papers have formalized systems similar to onion routing but without persistent circuits. Camenisch and Lysyanskaya [3] prove that the cryptography their protocol uses doesn't leak any information to nodes in the path other than the previous and next nodes, but leave open what anonymity it provides. This question is answered in part by Mauw et al. [10], who formalize a similar connectionless protocol in an ACP-style process algebra and, under a possibilistic definition of anonymity, show that it provides sender and receiver anonymity against a global passive adversary. Cryptography is dealt with using high level assumptions, similar to our work. Their model and analysis has much in common

with this paper, but it does differ in several important ways. First, the protocol they investigate is connectionless: each data "onion" stores the identities of the routers it will pass through. This is significantly different from onion routing, which is circuit-based. Second, the analysis is done with respect to a passive adversary, which exhibits only a subset of the behavior of an active adversary. Third, in their model agents choose destinations asynchronously and the adversary must take into account every onion he has seen when trying to break anonymity. In our model all agents choose a single destination, which gives the adversary more power. In some ways, our work extends theirs, and several of the differences noted here appear in [10] as suggestions for further research.

3 Model

3.1 Distributed system

Our model of onion routing is based on IO automata [9]. This formalism allows us to express an onion-routing protocol, model the network, and make precise the adversary's capabilities. One of its benefits is that it models asynchronous computation and communication. Another is that every action is performed by a single agent, so the perspective of the adversary is fairly clear.

We model onion routing as a fully connected asynchronous network of IO automata. The network is composed of FIFO channels. There is a set of users U and a set of routers R. Let $N = U \cup R$. The term *agent* refers to any element of N. It is possible that $U \cap R \neq \emptyset$. In this case, user and router automata exist on the same processor. We assume that the users all create circuits of a fixed length l. (In the current Tor network, $l = 3$.) Each router-and-user pair shares a set of secret keys; however, the router does not know which of its keys belong to which user. This separates, for now, key distribution from the rest of the protocol. We assume that all keys in the system are distinct. Let K be the keyspace. The triple (u, r, i) will refer to the ith key shared by user u and router r.

Let P be the set of control messages, and $\bar{P}$ be the extension of P by encryption with up to l keys. The control messages will be tagged with a link identifier and circuit identifier when sent, so let the protocol message space be $M = \mathbf{N}_+ \times \mathbf{N}_+ \times \bar{P}$. We denote the encryption of $p \in P$ using key k with $\{p\}_k$, and the decryption with $\{p\}_{-k}$. For brevity, the multiply encrypted message $\{\{p\}_{k_1}\}_{k_2}$ will be denoted $\{p\}_{k_1,k_2}$. Brackets will be used to indicate the list structure of a message (*i.e.* $[p_1, p_2, \ldots]$).

The adversary in our system is a set of users and routers $A \subseteq N$. The adversary is active in the sense that the automata running on members of A are completely arbitrary. We call an agent a *compromised* if $a \in A$

3.2 Automata

We give the automata descriptions for the users and routers that are based on the Tor protocol [6]. We have simplified the protocol in several ways. In particular

we do not perform key exchange, do not use a stream cipher, have each user construct exactly one circuit to one destination, do not include circuit teardowns, eliminate the final unencrypted message forward, and omit stream management and congestion control. We are also using circuit identifiers to mimic the effect of a timing attack. Section 4.7 discusses the effects of changing some of these features of our protocol.

During the protocol each user u iteratively constructs a circuit to his destination. u begins by sending the message $\{\text{CREATE}\}_{k_1}$ to the first router, r_1, on his circuit. The message is encrypted with a key, k_1, shared between u and r_1. r_1 identifies k_1 by repeatedly trying to decrypt the message with each one of its keys until the result is a valid control message. It responds with the message CREATED. (Note that this is different from the implemented Tor protocol, in which the CREATE message would be encrypted with the public key for r_1 rather than one of the shared keys it holds.) Given a partially-constructed circuit, u adds another router, r_i, to the end by sending the message $\{[\text{EXTEND}, r_i, \{\text{CREATE}\}_{k_i}]\}_{k_{i-1},\ldots,k_1}$ down the circuit. As the message gets forwarded down the circuit, each router decrypts it. r_{i-1} performs the CREATE steps described above, and then returns the message $\{\text{EXTENDED}\}_{k_{i-1}}$. Each router encrypts this message as it is sent back up the circuit.

Link identifiers are used by adjacent routers on a circuit to differentiate messages on different circuits. They are unique to the adjacent pair. Circuit identifiers are also included with each message and identify the circuit it is traveling on. They are unique among all circuits. Circuit identifiers are not used in the actual Tor protocol, and their only purpose here is to represent the ability of an adversary to insert and/or detect timing patterns in the traffic along a circuit. This reflects in our model the very real threat of timing attacks [11]. It has the added advantages of making it clear when this power is used and of being easy to remove in future model adjustments.

The user automaton's state consists of the sequence of routers in its circuit, a number that identifies its circuit, and a number that indicates the state of its circuit. We consider the final router in the circuit to be the destination of the user. The user automaton runs two threads, one to extend a circuit that is called upon receipt of a message and the other to start circuit creation that is called at the beginning of execution. We express these in pseudocode rather than IO automata, but note that the state changes in a particular branch occur simultaneously in the automaton. $k(u, c, b)$ refers to the key used by user u with router c_b in the bth position in circuit c. The automaton for user u appears in Automaton 1.

The router automaton's state is a set of keys and a table, T, with a row for each position the router holds in a circuit. Each row stores the previous and next hops in the circuit, identifying numbers for the incoming and outgoing links, and the associated key. There is only one thread and it is called upon receipt of a message. In the automaton for router r, we denote the smallest positive integer that is not being used on a link from r to q or from q to r as $minid(T, q)$. The automaton for router r appears in Automaton 2.

Automaton 1 User u

1: $c \in \{(r_1, \ldots, r_l) \in R^l | \forall_i r_i \neq r_{i+1}\}$; init: arbitrary ▷ User's circuit
2: $i \in \mathbf{N}$; init: random ▷ Circuit identifier
3: $b \in \mathbf{N}$; init: 0 ▷ Next hop to build
4: **procedure** START
5: SEND(c_1, $[i, 0, \{\text{CREATE}\}_{k(u,c,1)}]$)
6: $b = 1$
7: **end procedure**
8: **procedure** MESSAGE(msg,j) ▷ $msg \in M$ received from $j \in N$
9: **if** $j = c_1$ **then**
10: **if** $b = 1$ **then**
11: **if** $msg = [i, 0, \text{CREATED}]$ **then**
12: $b++$
13: SEND($c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{k(u,c,b)}]\}_{k(u,c,b-1),\ldots,k(u,c,1)}]$)
14: **end if**
15: **else if** $b < l$ **then**
16: **if** $msg = [i, 0, \{\text{EXTENDED}\}_{k(u,c,b-1),\ldots,k(u,c,1)}]$ **then**
17: $b++$
18: SEND($c_1, [i, 0, \{[\text{EXTEND}, c_b, \{\text{CREATE}\}_{k(u,c,b)}]\}_{k(u,c,b-1),\ldots,k(u,c,1)}]$)
19: **end if**
20: **else if** $b = l$ **then**
21: **if** $msg = [i, 0, \{\text{EXTENDED}\}_{k(u,c,b-1),\ldots,k(u,c,1)}]$ **then**
22: $b++$
23: **end if**
24: **end if**
25: **end if**
26: **end procedure**

3.3 System execution

We use standard notions of *execution* and *fairness*. An execution is a possible run of the network given its initial state. Fairness for us means that any message an automaton wants to send will eventually be sent and every sent message is eventually received.

We introduce the notion of a *cryptographic* execution. This is an execution in which no agent sends a control message encrypted with active keys it doesn't possess before it receives that message. We restrict our attention to such executions, and must require our encryption operation to prevent an attacker from outputting a control message, with more than negligible probability, when it is encrypted with keys he doesn't possess. This is reasonable because we can easily create a ciphertext space that is much larger than the rather limited control message space P. Note that this precludes the use of public key encryption to encrypt the packets because such messages can easily be constructed with the public keys of the routers.

Definition 1 *An* execution *is a sequence of states of an IO automaton alternating with actions of the automaton. It begins with an initial state, and two consecutive states are related by the automaton transition function and the action*

Automaton 2 Router r

$keys \subseteq K$, where $|keys| \geq |U| \cdot \lceil \frac{l}{2} \rceil$; init: arbitrary ▷ Private keys
$T \subset N \times \mathbf{N} \times R \times \mathbf{Z} \times keys$; init: $\emptyset$ ▷ Routing table
procedure MESSAGE($[i, n, p], q$) ▷ $[i, n, p] \in M$ received from $q \in N$
  if $[q, n, \emptyset, -1, k] \in T$ **then** ▷ In link created, out link absent
    if $\exists_{s \in R-r, b \in P} p = \{[\text{EXTEND}, s, b]\}_k$ **then**
      SEND($s, [minid(T, s), b]$)
      $T = T - [q, n, \emptyset, -1, k] + [q, n, s, -minid(T, s), k]$
    end if
  else if $[s, m, q, -n, k] \in T$ **then** ▷ In link created, out link initiated
    if $p = \text{CREATED}$ **then**
      $T = T - [s, m, q, -n, k] + [s, m, q, n, k]$
      SEND($s, [i, m, \{\text{EXTENDED}\}_k]$)
    end if
  else if $\exists_{m>0}[q, n, s, m, k] \in T$ **then** ▷ In and out links created
    SEND($s, [i, m, \{p\}_{-k}]$) ▷ Forward message down the circuit
  else if $[s, m, q, n, k] \in T$) **then** ▷ In and out links created
    SEND($s, [i, m, \{p\}_k]$) ▷ Forward message up the circuit
  else
    if $\exists_{k \in keys} p = \{\text{CREATE}\}_k$ **then** ▷ New link
      $T = T + [q, n, \emptyset, -1, k]$
      SEND($q, [i, n, \text{CREATED}]$)
    end if
  end if
end procedure

between them. Every action must be enabled*, meaning that the acting automaton must be in a state in which the action is possible at the point the action occurs.*

Definition 2 *A finite execution is* fair *if there are no actions enabled in the final state. Call an infinite execution* fair *if every output action that is enabled in infinitely many states occurs infinitely often.*

Definition 3 *An execution is* cryptographic *if an agent sends a message containing $\{p\}_{k_1,\ldots,k_i}$ only when it possesses all keys $k_1, \ldots, k_i$, or when for the largest j s.t. the agent does not possess k_j, $1 \leq j \leq i$, the agent has already received a message containing $\{p\}_{k_1,\ldots,k_j}$.*

3.4 Distinguishability

Definition 4 *A* configuration $C : U \to \{(r_1, \ldots, r_l, n) \in R^l \times \mathbf{N}_+ | \forall_i r_i \neq r_{i+1}\}$ *maps each user to the circuit and circuit identifier in his automaton state.*

The actions we want to be performed anonymously are closely related to the circuits the users try to construct during an execution. In our model, all messages are sent along links of a circuit; these messages are all circuit-creation messages and thus are entirely determined by the circuit, so the sender or receiver of a

given message corresponds directly to the path of the circuit. Therefore, in order to prove that certain actions are performed anonymously in the network, we can just show that the adversary can never determine this circuit information. This is a possibilistic notion of of anonymity. We do this by identifying classes of adversary-indistinguishable configurations.

Because $i \in N$ only sees those messages sent to and from i, an execution of a configuration C may appear the same to i as a similar execution of another configuration D that only differs from C in parts of the circuits that are not adjacent to i and in circuit identifiers that i never sees. To be assured that i will never notice a difference, we would like this to be true for all possible executions of C. These are the fair cryptographic executions of C, and likewise the executions of D should be fair and cryptographic. We will say that these configurations are indistinguishable if, for any fair cryptographic execution of C, there exists a fair cryptographic execution of D that appears identical to i, *i.e.* in which i sends and receives what appear to be the same messages in the same order.

Agent i's power to distinguish among executions is weakened by encryption in two ways. First, we allow a permutation on keys to be applied to the keys of encrypted or decrypted messages in an execution. This permutation can map a key from any router other than i to any other key of any other router other than i, because i can only tell that it doesn't hold these keys. It can map any key of i to any other key of i, because i doesn't know for which users and circuit positions its keys will be used. Second, i cannot distinguish among messages encrypted with a key he does not possess, so we allow a permutation to be applied to control messages that are encrypted with a key that is not shared with i. This second requirement must be justified by the computational intractability of distinguishing between encrypted messages with more than a negligible probability in our cryptosystem.

Definition 5 *Let D_A be a relation over configurations indicating which configurations are indistinguishable to $A \subseteq N$. For configurations C and C', $C \sim_{D_A} C'$ if for every fair cryptographic execution α of C, there exists some action sequence β s.t. the following conditions hold with C' as the initial state:*

1. *Every action of β is enabled, except possibly actions done by members of A.*
2. *β is fair for all agents, except possibly those in A.*
3. *β is cryptographic for all agents.*
4. *Let Ξ be the subset of permutations on the active keyspace $U \times R \times \lceil \frac{l}{2} \rceil$ s.t. each element restricted to keys involving $a \in A$ is a permutation on those keys. We apply $\xi \in \Xi$ to the encryption of a message sequence by changing every list component $\{p\}_{(u,r,i)}$ in the sequence to $\{p\}_{\xi(u,r,i)}$.*
   Let Π be the subset of permutations on $\bar{P}$ s.t. for all $\pi \in \Pi$, π is a permutation on the set $\{\{p\}_{k_1,\ldots,k_i}\}_{p \in P}$, and $\pi(\{p\}_{k_1,\ldots,k_i,k_a}) = \pi(\{p\}_{k_1,\ldots,k_i})$ when k_a is shared by the adversary. We apply $\pi \in \Pi$ to a message sequence by changing every message $\{p\}_{k_1,\ldots,k_i}$ in the message sequence to $\pi(\{p\}_{k_1,\ldots,k_i})$.

Then there must exist $\xi \in \Xi$ and $\pi \in \Pi$ s.t. applying ξ and π to the subsequence of α corresponding to actions of A yields the subsequence of β corresponding to actions of A.

If $C \sim_{D_A} C'$, we say that C is *indistinguishable* from C' to A. It is clear that an indistinguishability relation is reflexive and transitive.

3.5 Anonymity and Unlinkability

The sender in this model corresponds to the user of a circuit, the receiver to the last router of the circuit, and the messages we wish to communicate anonymously are just the circuit control messages. The circuit identifiers allow the adversary to link together all the messages initiated by a user and attribute them to a single source. (Recall that in our model, users open a single circuit to a unique destination at one time.) Therefore sender anonymity is provided to u if the adversary can't determine which circuit identifier u is using. Similarly, receiver anonymity is provided to r for messages from u if the adversary can't determine the destination of the circuit with u's identifier. Also, unlinkability is provided to u and r if the adversary can't determine u's destination.

Definition 6 *User u has* sender anonymity *in configuration C with respect to adversary A if there exists some indistinguishable configuration C' in which u uses a different circuit identifier.*

Definition 7 *Router r has* receiver anonymity *on user u's circuit, in configuration C, and with respect to adversary A, if there exists some indistinguishable configuration C' in which a user with u's circuit identifier, if one exists, has a destination other than r.*

Definition 8 *User u and router r are* unlinkable *in configuration C if there is some indistinguishable configuration C' in which the destination of u is not r.*

4 Indistinguishable Configurations

Now we will show that sometimes the adversary cannot determine the path or identifier of a circuit. More specifically, an adversary can only determine which user or router occupies a given position in a circuit when the adversary controls it or a router adjacent to it on that circuit. Also, when the adversary controls no part of a circuit it cannot determine its identifier.

In order to do this, we must show that, given a pair of configurations (C, C') that are indistinguishable by these criteria, for every execution of C there exists an execution of C' that appears identical to the adversary. To do this we will start with a fair cryptographic execution of C, describe how to transform it, and prove that this transformed sequence forms a fair, cryptographic, and indistinguishable execution of C'. We will also show that a pair of configurations that are distinguishable by the described criteria allow no such transformation.

4.1 Message Sequences

To start, we observe that, in spite of the arbitrary actions of the adversary, the actions of the uncompromised users and routers in an execution are very structured. The protocol followed by the user and router automata defines a simple sequence of message sends and receives for every circuit. A user or router will only send a message as part of such a sequence.

The user subsequence consists of messages between the user and the first router on its circuit. The user u in configuration C begins the sequence by sending a CREATE message to $C_1(u)$, the first router in u's circuit in C; $C_1(u)$ responds with a CREATED message. Then for the remaining $l-1$ routers on the circuit, the user sends an EXTEND message and $C_1(u)$ responds with an EXTENDED message. The user will only send a message as the next step in this sequence. Therefore we can take all actions in an execution by u and partition them into those that are part of this sequence and those that are not. Those that are not are "junk" receives that the adversary caused to be sent to u by not following the protocol.

A router performs a similar sequence when it is added to a circuit. The sequence begins when router r router receives a CREATE message from agent n with the smallest unused link identifier at that point between r and n. r responds with a CREATED message. Then r receives an EXTEND message with the identity of the next router q and an enclosed CREATE message. It passes the enclosed message on to q, receives a CREATED message back, and sends an EXTENDED message to n. Then r forwards up or down the circuit any further messages received. r will only send messages as part of such a sequence. We can therefore partition all actions by r in an execution into sequences of this type, in addition to a sequence for "junk" receives that aren't part of such a sequence and are a result of adversarial misbehavior. We will use the existence of such partitions of executions in our analysis.

4.2 Indistinguishable Users

Now we prove that an active adversary cannot determine which user creates a given circuit unless the first router on that circuit is controlled by the adversary or the owners of all the other circuits have been determined. That is, an adversary cannot distinguish between a configuration C and the configuration C' that is identical to C except for two circuits with uncompromised first routers that are switched between the circuit owners. In order to do so, we must show that, for any fair cryptographic execution of C, there exists some action sequence of C' satisfying the indistinguishability requirements of Definition 5. To do so, we simply swap between the switched users the messages that pass between them and the first routers on their circuits and switch the encryption keys of these messages.

Theorem 1 *Let u, v be two distinct users s.t. neither they nor the first routers in their circuits are compromised (*i.e.*, are in A). Let C' be identical to C except the circuits of users u and v are switched. C is indistinguishable from C' to A.*

Proof Sketch: Let α be a fair cryptographic execution of C. To create a possible execution of C', first construct α' by replacing any message sent or received between u (v) and $C_1(u)$ ($C_1(v)$) in α with a message sent or received between v (u) and $C_1(u)$ ($C_1(v)$). Then let ξ be the permutation that sends u to v and v to u and other users to themselves. Create β by applying ξ to the encryption keys of α'.

To show that the actions in this sequence are enabled for uncompromised routers we observe that only message partitions for u, v, $C_1(u)$, and $C_1(v)$ have been changed. These are modified so that they remain valid partitions,. To show that the execution is fair we observe that no "new" valid messages or sequences can appear. The transformed sequence is cryptographic because the key permutations and message changes are applied to the entire sequence and the original sequence α was cryptographic. The permutation needed to make β look like α to A is just the reverse of the key permutation used to create β. □

4.3 Indistinguishable Routers

Now we prove that an adversary cannot determine an uncompromised router on a given circuit unless it controls the previous or next router on that circuit. More formally, assume that the $(i-1)$st, ith, and $(i+1)$st routers of a user u's circuit in some configuration C are not compromised. We will show that C is indistinguishable from configuration C', where C' is identical to C except the ith router of u's circuit has been arbitrarily changed. The proof is similar to that of Theorem 1, although it is complicated by the fact that the identities of routers in a circuit are included in multiple ways in the circuit creation protocol. Given an execution of C, we identify those message that are part of the circuit creation sequence of the modified circuit and then change them to add a different router in the ith position. Then we show that, in the sense of Definition 5, from the adversary's perspective this sequence is indistinguishable from the original and could be an execution of C'.

Theorem 2 *Say there is some user $u \notin A$ s.t. u's circuit in C contains three consecutive routers, $r_{i-1}, r_i, r_{i+1} \notin A$. Let C' be equal to C, except r_i is replaced with r'_i in u's circuit, where $r'_i \notin A \cup \{r_{i-1}, r_{i+1}\}$. C' is indistinguishable from C to A. The same holds for uncompromised routers (r_i, r_{i+1}) if they begin u's circuit and are replaced with (r'_i, r_{i+1}), or (r_{i-1}, r_i) if they end u's circuit and are replaced with (r_{i-1}, r'_i).*

Proof Sketch: Let α be some fair cryptographic execution of C. Let $h(C(u), i)$ be the number of occurrences of the ith router in the circuit $C(u)$ among the first i routers. We modify α in steps to create an indistinguishable sequence β:

1. Replace all messages of the form $[\text{EXTEND}, r_i, \{\text{CREATE}\}_{u,r_i,h(C(u),i)}]$ with $[\text{EXTEND}, r'_i, \{\text{CREATE}\}_{u,r'_i,h(C'(u),i)}]$.
2. Consider the partitions of router r_{i-1}'s actions that each form a prefix of the sequence adding r_{i-1} to u's circuit as the $(i-1)$st router. Replace all

messages in these partitions that are to and from r_i with the same messages to and from r'_i. Modify the link identifiers on these messages so that they are the smallest identifiers in use between r_{i-1} and r'_i at that point in α. Increase link identifiers that are in use between r_{i-1} and r'_i to make room for these new connections and decrease link identifiers that are in use between r_{i-1} and r_i to fill in the holes created by the removed connections. Perform similar modifications for routers r_i and r_{i+1}.

3. Replace all keys of the form $(u, r_i, h(C(u), i))$ with $(u, r'_i, h(C'(u), i))$. Increment as necessary the third component of the encryption keys used between u and r'_i to take into account that r'_i appears once more in $C'(u)$ than it does in $C(u)$. Also decrement as necessary the third component of the keys used between u and r_i to take into account that r_i appears once less in $C'(u)$ than it does in $C(u)$.

The actions in the transformed sequence are enabled because we convert the partitions involving r_i to involve r'_i instead, adjusting link and key numbering as needed to maintain global consistency. β is cryptographic because the key and message permutations used to create it are applied uniformly. The transformed execution is fair first because we modify partitions as a whole, so there are no partial unfinished sequences. Second, β is cryptographic, so no "junk" receives from the adversary could be valid messages in a transformed partition. An attack that under this reasoning can't occur is that a compromised router a can't send a valid EXTEND message directing the router r'_i at the end of u's partially-constructed circuit to create a link to a. Such a message, if a weren't prohibited from sending it, only enables router action when r'_i is the router at the end of the circuit, since another router would be using a different key. This would leave r'_i with an enabled action in β. Finally, the required permutations to make β appear like α to A are simply the reverse of those used to create β in the first place. □

4.4 Indistinguishable Identifiers

Theorem 3 *Say there is some uncompromised user u s.t. all routers in $C(u)$ are uncompromised. Let C' be a configuration that is identical to C, except that u uses a different circuit identifier. C' is indistinguishable from C to A.*

Proof. Let α be a fair cryptographic execution of C. To create β, simply change every occurrence of u's circuit identifier in C ($C_{l+1}(u)$) to its identifier in C'. β is enabled, fair, and cryptographic for C' because no message containing $C_{l+1}(u)$ gets sent to the adversary in α and the protocol itself ignores circuit identifiers except to forward them on. It appears the same to A for the same reason.

4.5 Distinguishable Configurations

The relation D_A, when restricted to the transitive closure of pairs that are indistinguishable by Theorems 1-3, is symmetric, and therefore forms an equivalence relation. Let $\approx_{D_A}$ denote this relation.

We can easily tell which configurations are in the same equivalence class using a function ρ that reduces a circuit to an identifier, the compromised positions, and the positions adjacent to compromised positions. For convenience, in the following we take c_0 to refer to u.

Definition 9 *Let* $\rho : U \times N^l \times \mathbf{N}_+ \times \mathcal{P}(N) \to \mathbf{N} \times \mathcal{P}(N \times \mathbf{N}_+)$ *be:*

$$\rho(u, c, A) = \begin{cases} (c_{l+1}, \{(r, i) \in N \times \mathbf{N}_+ | c_i = r \wedge (c_{i-1} \in A \vee c_i \in A \vee c_{i+1} \in A)\}) \\ \qquad\qquad \text{if } c_i \in A \text{ for some } i \\ (0, \emptyset) \qquad\qquad \text{otherwise} \end{cases}$$

We overload this notation: $\rho(C)$ refers to the multiset formed from the circuits of configuration C adjoined with their user and reduced by ρ, *i.e.*, $\rho(C) = \{\rho(u, C(u), A) | u \in U\}$. Thus ρ captures the indistinguishable features of a configuration according to Theorems 1, 2, and 3.

Now we show that the equivalence relation is in fact the entire indistinguishability relation and that Theorems 1, 2, and 3 characterize which configurations are indistinguishable.

Theorem 4 *If* $C \sim_{D_A} D$ *then* $C \approx_{D_A} D$.

Proof. We show the contrapositive. Suppose C and D are in different equivalence classes. Let the adversary run the automata prescribed by the protocol on the agents it controls. Let α and β be fair cryptographic executions of C and D respectively.

Partition the adversary actions of α into subsequences that share the same circuit identifier. There is at most one such partition for each circuit. Circuit positions that are created in the same partition belong to the same circuit. In each partition the adversary can determine the absolute location of a circuit position filled by a given compromised agent a by counting the total number of messages it sees after the initial CREATE. Clearly A can also determine the agents that precede and succeed a on the circuit and the circuit identifier itself. Therefore A can determine the reduced circuit structure $\rho(C)$ from α.

The adversary can use β in the same way to determine $\rho(D)$. It is easy to see that $C \approx_{D_A} D$ if and only if $\rho(C) = \rho(D)$, so $\rho(C) \neq \rho(D)$. Therefore A can always distinguish between C and D.

4.6 Anonymity

The configurations that provide sender anonymity, receiver anonymity, and unlinkability follow easily from Theorems 1, 2, 3, and 4. In the following, let u be a user, C be a configuration, r be a router, and A be the adversary.

Corollary 1 *u has sender anonymity in C with respect to A if and only if at least one of two cases holds. The first is that u and $C_1(u)$ are uncompromised and there exists another user $v \neq u$ s.t. v and $C_1(v)$ are uncompromised. The second is that u and all $C_i(u)$ are uncompromised.* □

Corollary 2 *r has receiver anonymity on u's circuit in C with respect to A if and only if at least one of two cases holds. The first is that u, r, and $C_{l-1}(u)$ are uncompromised and there exists another router $q \neq r$ s.t. q is uncompromised. The second is that u and all $C_i(u)$ are uncompromised.* □

Corollary 3 *u and r are unlinkable in C with respect to A if and only if at least one of two cases holds. The first is that u, r, and $C_{l-1}(u)$ are uncompromised, and there exists another router $q \neq r$ s.t. q is uncompromised. For the second case it must be that u and $C_1(u)$ are uncompromised and there exists another user $v \neq u$ s.t. v and $C_1(v)$ are uncompromised. Also, it must be that $C_l(v) \neq r$, or $C_{l-1}(v)$ and r are uncompromised and there exists another router $q \neq r$ s.t. q is uncompromised.* □

4.7 Model Changes

We chose the described protocol to balance two goals. The first was to accurately model Tor. The second was to make it simple enough to be analyzed, and so that the main ideas of the analysis weren't unnecessarily complicated. Our results are robust to changes of the protocol, however. We can make the protocol simpler by removing multiple encryption and the circuit identifiers without weakening the indistinguishability results. Single encryption does allow the adversary to easily link entries in his routers by sending messages along the circuit. This power is already available in our model from circuit identifiers, though. In the other direction, we can make it more complicated with a stream cipher and multiple circuits per user without weakening the distinguishability results.

Stream ciphers are used in the Tor protocol and prevent signaling along a circuit using dummy messages. Sending such messages will throw off the counter by some routers on the circuit and the circuit will stop working. We can model a stream cipher by expressing the encryption of the ith message p with key k as $\{p\}_{(k,i)}$, and allowing a different permutation to be applied for every pair (k, i). This can only increase the size of the configuration indistinguishability relation. However, the proof for the distinguishability of configurations only relies on the ability of the adversary to decrypt using his keys, count messages, and recognize the circuit identifier. Therefore it still holds when the model uses a stream cipher.

Allowing users to create multiple circuits doesn't weaken the adversary's power to link together its circuit positions and determine their position, but the number of configurations that are consistent with this view does in some cases increase. Let users create an arbitrary number of circuits. The adversary can still link positions and count messages as before, so the adversary can distinguish configurations C and D if $\rho(C) \neq \rho(D)$. However, Theorems 1-3 no longer identify all indistinguishable configurations, because a circuit with an unknown user can belong to any user, not just a user with an uncompromised first router. It can, however, be shown that the converse of Theorem 4 continues to hold if we replace "$C \approx_{D_A} D$" with "$\rho(C) = \rho(D)$."

5 Conclusions

We have presented a model of onion routing and characterized when anonymity and unlinkability are provided. It is an asynchronous model using IO automata, and the protocol is based on Tor. The adversary we analyze is local and active in the sense that he is allowed to run arbitrary automata but is limited to the view of a subset of users and routers that he controls. We show that the adversary can determine when his routers hold positions in the same circuit and where in the circuit they are located, and only this. Thus anonymity is generally provided when the first or last circuit router is uncompromised.

Two directions for future work on modeling onion routing are improving the model and improving the analysis. A big missing piece in the current model is the lack of time. Timing attacks are successful in practice, and we have only approximated them in our model with circuit identifiers. Also, we have simplified the Tor protocol by omitting key exchange, circuit teardowns, the final unencrypted message forward, and stream management and congestion control. Adding some or all of these would bring the model closer to reality. Towards improving the analysis, we have made several assumptions about the cryptosystem but have not exhibited an encryption scheme for which they hold. Also, probabilities in both the behavior of the users and the operation of system should be added to the model and analyzed according to probabilistic definitions of anonymity.

References

1. Anonymity bibliography. http://freehaven.net/anonbib/.
2. P. Boucher, A. Shostack, and I. Goldberg. Freedom systems 2.0 architecture. White paper, Zero Knowledge Systems, Inc., 2000.
3. J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In *CRYPTO*, pages 169–187, 2005.
4. D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *CACM*, 28(10), October 1985.
5. C. Díaz, S. Seys, J. Claessens, and B. Preneel. Towards measuring anonymity. In *Privacy Enhancing Technologies*, 2002.
6. R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium*, 2004.
7. I. Goldberg. On the security of the Tor authentication protocol. In *Privacy Enhancing Technologies*, 2006.
8. D. Goldschlag, M. Reed, and P. Syverson. Hiding routing information. In *First International Workshop on Information Hiding*, pages 137–150, 1996.
9. Nancy Lynch. *Distributed Algorithms*. Morgan Kaufmann, first edition, 1996.
10. S. Mauw, J. Verschuren, and E. de Vink. A formalization of anonymity and onion routing. In *European Symposium on Research in Computer Security*, 2004.
11. L. Øverlier and P. Syverson. Locating hidden servers. In *IEEE Symposium on Security and Privacy*, 2006.
12. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In *Privacy Enhancing Technologies*, 2002.
13. P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In *Designing Privacy Enhancing Technologies*, pages 96–114, 2000.