A Formalization of Anonymity and Onion Routing

S. Mauw[1], J.H.S. Verschuren[1,2] and E.P. de Vink[1,3]

[1] Dept of Math. and Comp. Sc., Technische Universiteit Eindhoven
P.O. Box 513, 5600 MB Eindhoven, the Netherlands
[2] TNO ITSEF, P.O. Box 96864, 2509 JG 's-Gravenhage, the Netherlands
[3] LIACS, Leiden University, Niels Bohrweg 1, 2333 CA Leiden, the Netherlands

Abstract. The use of formal methods to verify security protocols with respect to secrecy and authentication has become standard practice. In contrast, the formalization of other security goals, such as privacy, has received less attention. Due to the increasing importance of privacy in the current society, formal methods will also become indispensable in this area. Therefore, we propose a formal definition of the notion of anonymity in presence of an observing intruder. We validate this definition by analyzing a well-known anonymity preserving protocol, viz. onion routing.

1 Introduction

Nowadays there is a growing concern about one's privacy. The adoption of techniques like RFID and DRM may have severe consequences for the privacy of the individual [8, 6]. The widespread acceptance of electronic services, such as location based services, electronic tolling, loyalty schemes, may carry consequences on the user's privacy. As 'privacy' is becoming more of an issue, there is an increasing need for analysis of systems in relation to privacy requirements.

The so-called functional class in the Common Criteria (CC, [10]) distinguishes between four aspects of privacy: anonymity, pseudonymity, unlinkability and unobservability. Anonymity, which is the topic of our current research, ensures that a subject may use a resource or service without disclosing its user identity. Pseudonymity ensures that a user may use a resource or service without disclosing its identity, but can still be accountable for that use. Unlinkability ensures that a user may make multiple uses of resources or services without others being able to link these uses together. Unlinkability differs from pseudonymity in the sense that, although in pseudonymity the user is also not known, relations between different actions can be provided. Unobservability ensures that a user may use a resource or service without others, especially third parties, being able to observe that the resource or service is being used.

Such informal definitions are essential to the understanding of the different notions of privacy, but will only allow to investigate a system informally. However, in contrast to other security properties, such as confidentiality and

authentication (see e.g. [14, 4]), privacy has hardly been studied from a formal methods point of view (see e.g. [12, 11, 3] for more or less informal approaches to anonymity). It is our aim to provide an appropriate formal definition of anonymity and validate this definition by analyzing the well-known onion routing protocol. In [14] a start is made to give a formal description of anonymity.

Our definition of anonymity is based on the above mentioned definition in the CC and on the definition of anonymity provided by Pfitzmann et al. [12], which reads "Anonymity is the state of being not identifiable within a set of subjects, the anonymity set." This anonymity group forms the basis of our definition. We say that a user u' is in the anonymity group of user u, if for every behaviour of the system that can be attributed to u, there is another possible behaviour of the system that can be attributed to u', such that an observer or intruder cannot tell the difference between these two behaviours. This means that an intruder cannot tell the difference between u and any other user in its anonymity group.

Onion routing was originally devised by Goldschlag, Reed, Syverson in [9, 17] as a solution for anonymous connections. Onion routing creates a layered data structure called an onion. As the data passes through each onion router along the way, one layer of encryption is removed according to the recipe contained in the onion. The Naval Research Lab has a test bed Onion Routing Network that is available for any one to use. While in operation, users in more than sixty countries initiated up to 1.5 million connections per month through the prototype system. This demand certainly shows an interest in the service. It also shows that it is feasible. Based on this success, a design for a second generation system was initiated [16].

Syverson et al. have performed an analysis of the second generation system for onion routing. In [16] different attacker models are considered, viz. single, multiple, roving and global adversary. A single and a multiple adversary point to a situation where only one core onion router, respectively, more core onion routers are compromised. In both cases the compromised onion routers are fixed. A roving adversary points to a situation where a fixed-bound size subset of core onion routers is compromised at any one time. At specific intervals, other core onion routers can become compromised or uncompromised. Syverson et al. rule out the global adversary (all core onion routers are compromised) as the onion routing infrastructure cannot realize any anonymity in that case. They compare the results with the protection provided by the Crowds model [13]. It is shown, that onion routing generally resists traffic analysis more effectively than any other published and deployed mechanisms for Internet communication.

Diaz et al. [5] and also Serjantov and Danezis [15] propose an information theoretic approach to measure the level of anonymity of a system. In their model the attacker will carry out a probabilistic attack: after observing the system, an attacker may assign some probabilities to each sender as being the originator of a message. This can be based on information the system is leaking, message lengths, traffic analysis, etc. Subsequently, the entropy is used as a tool to calculate the degree of anonymity achieved by the users of a system towards a

particular attacker. Their measurement method is applied to analyze the degree of anonymity of crowds and onion routing.

We provide a possibilistic analysis of the onion routing protocol in a process algebraic framework. We aim at determining the anonymity groups of part taking users under different circumstances, such as conspiring routers. In order to appreciate the intricacies of the onion routing protocol, we have analyzed several weaker protocols too. In this paper we will only report on our findings with respect to a variation which we coined coconut routing.

Our paper is structured as follows. In Section 2 we provide a formal definition of anonymity in a trace model. In Section 3 we explain an abstraction of the onion routing protocol and give its formal specification in process algebra. We also provide an alternative characterization which will show helpful in the formal analysis of Section 4. In Section 5 we discuss conclusions and future research.

2 Formal definitions

Intruder capabilities In this section we define the notion of anonymity in presence of an eavesdropping intruder. In general, the intruder can overhear the communication, but can not interpret all the messages. Dependent on the set of keys known to the intruder, some behaviours of the system observed by the intruder can be distinguished while others can not. If the intruder can not, based on its eavesdropping capacities, distinguish between two users, these two users are in the same anonymity group.

We fix a set of users $\mathcal{U}$, a set of actions $\mathcal{A}$, a set of traces $\mathcal{T}$ and a set of keys $\mathcal{K}$. The set of actions $\mathcal{A}$ is split up into a subset of observable actions $\mathcal{A}_{\text{obs}}$ and a set of invisible actions $\mathcal{A}_{\text{inv}}$. For our purposes, the set of traces $\mathcal{T}$ consists of all finite and infinite words of actions from $\mathcal{A}$.

The actions come equipped with some syntactic structure, the details of which do not matter here. We assume that the set $\mathcal{A}$ can be viewed as the collection of terms of some signature that includes an operation $e, k \mapsto \{e\}_k$ for an expression e and key k. Let $K \subseteq \mathcal{K}$ be a set of keys. A tagging function $\theta\colon \mathcal{A} \to \mathcal{A}_\Theta$, with Θ some fixed set of tags, maps actions in $\mathcal{A}$, i.e. terms over the implicit signature, to tagged actions in $\mathcal{A}_\Theta$, i.e. terms over the implicit signature extended with the elements of Θ as new constants. The function θ is assumed to be injective; the idea is to replace undecryptable subterms by some tag, such that equal subterms are identified with the same tag. More concretely, the mapping $\theta_K\colon \mathcal{A} \to \mathcal{A}_\Theta$ has the property $\theta_K(\{e\}_k) = \{\theta_K(e)\}_k$ if the key k is in the set of keys K and $\theta_K(\{e\}_k) = \theta(\{e\}_k)$ if not. The mapping θ_K extends naturally from actions to traces.

We say that two traces $t_1, t_2 \in \mathcal{T}$ are K-equivalent, notation $t_1 \sim_K t_2$, if, for some bijection $\beta\colon \Theta \to \Theta$, it holds that $\theta_K(t_1) = (\beta \circ \theta)_K(t_2)$. The interpretation of $t_1 \sim_K t_2$ is that the traces t_1 and t_2 are equal for what can be observed, possibly after decryption with the keys in K, and there is a global correspondence between the parts that can not be decrypted. Suppose we have the actions $a = \{e_1\}_k$, $b = \{e_2\}_k$ and $c = \{e_3\}_k$, and actions $x = \{e\}_{k_1}$, $y = \{e\}_{k_2}$ and $z = \{e\}_{k_3}$.

The tagging of traces is a means to express the capability of an intruder to distinguish, e.g., the 2-element traces $a \cdot b$ and $c \cdot c$ even if the intruder can not decrypt any of the terms . Likewise, we have $x \cdot y \not\sim_K z \cdot z$ irrespective of which of the keys k_1, k_2 and k_3 are known.

The above notion of tagging and renaming captures that not all information carried by a trace can be distinguished by the intruder. This even stretches a little further: some actions will not be visible at all. Although we assume, that the intruder can overhear all network activity, the internal activity of principals can not be observed. Therefore, we have means to restrict a trace t over $\mathcal{A}$ to a trace over $\mathcal{A}_{\text{obs}}$ of observable actions. The mapping $obs\colon \mathcal{T} \to \mathcal{T}$ is such that $obs(\varepsilon) = \varepsilon$, $obs(a \cdot t) = a \cdot obs(t)$ if $a \in \mathcal{A}_{\text{obs}}$ and $obs(a \cdot t) = obs(t)$ if not. We use the notation $t_1 \sim_{obs}^{K} t_2$ iff $obs(t_1) \sim_K obs(t_2)$. When the set of keys K is clear from the context, we simply write $t_1 \sim_{obs} t_2$. For the sake of simplicity, we treat K as a constant set of keys. This will suffice for the treatment of the onion routing protocol, where the intruder does not learn any new keys during operation. In general, however, a privacy protocol may leak encryption keys, requiring a dynamic modeling of K. This extension is rather straightforward and will not influence the main line of reasoning. Therefore, we will ignore this possibility.

User attribution Next, we address the notion of attributing a trace to a user. As a trace can contain interaction of various sessions of many principals in different roles, we introduce a mechanism to focus on a particular action of a trace. In concrete situations we fix a so-called attribution function for each role that is of interest. Such an attribution function $\alpha\colon \mathcal{T} \times \mathbb{N} \to \mathcal{U}$ returns the user in the particular role, involved in the interaction corresponding to the n-th action $t[n]$ of the trace t. For example, in a four step key agreement protocol we can distinguish between the roles of initiator, responder and server. A trace t of a system with many users, acting both as initiator and responder, contains many sessions. There may be some communication at position n of t corresponding to the third step of the protocol. The attribution function for initiator then returns the initiator of the particular protocol session; the attribution function for the responder returns the responder of the particular session.

The attribution function $\alpha(\cdot, \cdot)$ does not take the intruder into account. In general, the intruder considers a particular occurrence of an action in a trace or part of a trace and tries to identify the user or users involved in this. However, the selection by the intruder of the action of interest is based on observable actions only. If two traces are observationally the same to the intruder, its analysis is focused on the same action. The traces generally differ in the number and/or position of invisible actions, so that the particular action can be at different positions in the two traces. Therefore, we introduce the partial mapping $obscnt\colon \mathcal{T} \times \mathbb{N} \xrightarrow{p} \mathbb{N}$, that returns for a position n in a trace t the corresponding position $obscnt(t, n)$ in the reduced trace, i.e. $obscnt(t, 0) = 0$, $obscnt(a \cdot t, n + 1) = obscnt(t, n) + 1$ if $a \in \mathcal{A}_{\text{obs}}$ and $obscnt(a \cdot t, n + 1) = obscnt(t, n) + 1$ if $a \notin \mathcal{A}_{\text{obs}}$, and $obscnt(\varepsilon, n + 1)$ is undefined.

Selection functions Next, we introduce selection functions, that take only observables and relative positions of observables into account. A selection function reflects a specific preference of the intruder. Formally, a mapping $\sigma\colon \mathcal{T} \to \mathbb{N}$ is called a selection function, with respect to a set of keys K, if (i) $\sigma(t) \in dom(t)$, (ii) $t[\sigma(t)] \in \mathcal{A}_{\text{obs}}$, and (iii) if $t_1 \sim_{obs}^{K} t_2$ then $obscnt(t_1, \sigma(t_1)) = obscnt(t_2, \sigma(t_2))$. Thus, if traces t_1 and t_2 are the same to the intruder that knows about the keys in K, then the selection function σ points in t_1 and in t_2 to an observable action at corresponding positions. Given the choice of positions governed by a selection function σ, the attribution α_σ of users to a trace, induced by the selection function σ, is then simply defined as $\alpha_\sigma(t) = \alpha(t, \sigma(t))$. Note that, in general, we do not have $\alpha(t_1, \sigma(t_1)) = \alpha(t_2, \sigma(t_2))$.

Below, we have occasion to further restrict the selection function that we consider. Intuitively, it is clear that an intruder observing a system from its very first startup is more powerful, than an intruder viewing the same system from some moment in time onwards. Typically, we assume the selection functions to stem from a class Σ of selection functions that point at positions in traces beyond some initialization phase. Likewise, we only want to consider the traces of the system under consideration. Thus, for a given system $\mathcal{S}$ with the set of traces $Tr(\mathcal{S})$ as its behaviour, we restrict the choice of traces to $Tr(\mathcal{S})$ or a subset thereof (for example, fair traces). Thus, a selection function σ will have functionality $\sigma\colon Tr(\mathcal{S}) \to \mathcal{U}$ rather than $\sigma\colon \mathcal{T} \to \mathcal{U}$.

Anonymity With this in mind, we are ready for the definition of an anonymity group of a user. For a user u, its anonymity group $AG(u)$ consists of all users u' that can not be distinguished from u by the intruder: u' is in the anonymity group of u with respect to an attribution α and selection function σ, if for any trace t that is attributed to u, i.e. $\alpha(t, \sigma(t)) = u$, we can find an observationally equivalent trace t', i.e. $t' \sim_{obs} t$, that is attributed to u'. So, given the observable behavior one can not tell whether u or any other user u' in its anonymity group was involved.

Definition 1. *Let $\mathcal{S}$ be a system, $\mathcal{A}_{\text{obs}} \subseteq \mathcal{A}$ a set of observable actions, $K \subseteq \mathcal{K}$ a set of keys, Σ a class of selection functions and α an attribution function. For a user $u \in \mathcal{U}$, its anonymity group $AG(u)$ is given by*

$$AG(u) = \{\, u' \in \mathcal{U} \mid \forall \sigma \in \Sigma\, \forall t \in Tr(\mathcal{S})\, \exists t' \in Tr(\mathcal{S})\colon \\ \alpha_\sigma(t) = u \to \alpha_\sigma(t') = u' \wedge t \sim_{obs} t' \,\}.$$

Clearly, the size of $AG(u)$ is an indication for the degree of anonymity user u has. Furthermore, note that, in general, we do not have $v \in AG(u) \Leftrightarrow u \in AG(v)$, so the set of anonymity groups does not form a partition of the set of users. In Section 4, we will exploit our definition of anonymity in a formal analysis of the onion routing protocol.

3 An example: onion routing

We discuss the onion routing protocol for illustrating the formalization of privacy. After an informal explanation of the onion routing protocol, we provide

a formal specification in ACP-style process algebra. We will also briefly discuss a weaker security protocol, which we call coconut routing. Next, we present an alternative characterization of the onion routing protocol which helps in proving our anonymity results in Section 4.

3.1 The onion routing protocol

The onion routing protocol as devised by Syverson et al. [17] is a constellation of measures to establish anonymous connections between two agents. It is our intention to formally describe and analyze the smallest protocol based on the onion routing principle that still exhibits interesting privacy properties. Starting point is a network of routers. We assume that the network forms a connected graph, which means that there is a path from every router to every other router. Such a path may be comprised of a series of intermediate routers.

To every router we associate a collection of users. Connections between a user and its router are typically realized within a local network and will be considered secure, while connections between two routers are not controlled by either router and may belong to a global communication infrastructure such as the Internet. It is realistic to assume that remote routers and connections between routers may be compromised. Given this possibly hostile environment, the purpose of the onion routing protocol is to enable a user S to send a message to a user R without revealing the identity of S nor that of R.
In order to establish the above requirement, we assume the existence of a public key infrastructure for the routers. This means that every router (whether compromised or not) has a public/private key pair and that all routers know the public keys of all other routers. The Message Sequence Chart of Figure 1 explains how the protocol operates. Suppose that user S intends to send message m to user R. For S this simply means that it uses its router OS as a proxy to perform this task. Therefore, it sends message m and recipient R to OS. Next, its router determines a path leading to the router to which R belongs and packs the message in such a way that every node in the path can only deduce the next node in the path, but nothing else. Suppose the chosen path is $OS; O1; O2; OR$, then the message sent to $O1$ is $O1, \{O2, \{OR, \{R, m\}_{pk(OR)}\}_{pk(O2)}\}_{pk(O1)}$, i.e. a header identifying the intended intermediate recipient $O1$ and a payload of some content encrypted with the public key of $O1$. Since we expect that $O1$ only knows its own secret key, $O1$ can only peel off the outermost layer of this composite message. Therefore, $O1$ obtains message $O2, \{OR, \{R, m\}_{pk(OR)}\}_{pk(O2)}$ and learns that this message has to be passed through to router $O2$. Likewise, $O2$ and OR peel off their layer from the onion and, finally, OR knows that it has to send message m to its user R.

The reason why this protocol establishes privacy of the sender and receiver of a message lies in the fact that the messages leaving a router cannot be related to the messages that have entered a router. This unlinkability of incoming and outgoing messages requires that an attacker cannot trace an outgoing message back to an incoming message by simply trying the public keys of all routers. Therefore, we require randomized encryption, which means that the same message

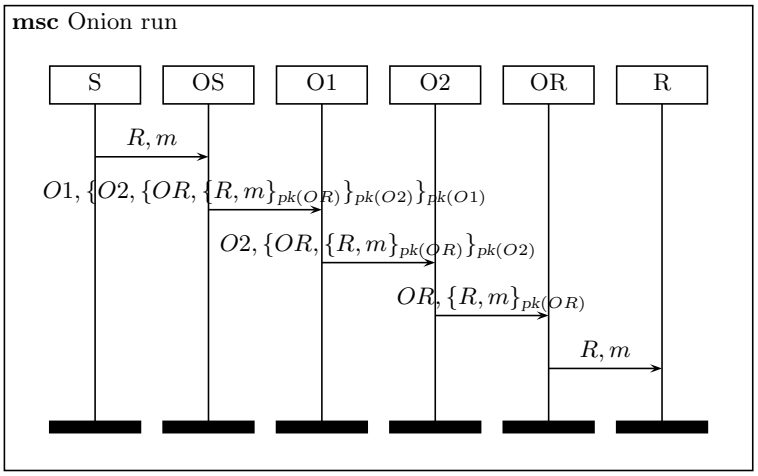

Fig. 1. Sample run of the onion routing protocol.

encrypted with the same key every time yields a different enciphered message. This can be established e.g. by salting the input. Which conditions exactly guarantee which kind of privacy is subject of the formal analysis later in this paper.

3.2 Coconut routing

The onion routing protocol works because the messages are packed in a series of shells, which are subsequently peeled off by the conveying routers. It is interesting to study weaker variants of this protocol and see how onion routing solves the weaknesses. To this end, we introduce a variation on onion routing, which we will call *coconut routing*. We will only conduct an informal analysis of this weaker protocol. A thorough analysis follows along the same steps as the analysis of the onion routing protocol above.

In the coconut routing protocol, the original message and the path are encrypted with a symmetric cryptographic key. This key is a secret shared by all routers. Figure 2 shows a sample run of the coconut routing protocol and suffices to understand its operation.

3.3 Formal specification of onion routing

The above describes onion routing informally. Next, we define the onion routing protocol in ACP-style process algebra (see, e.g., [1, 7, 2]). We assume that the reader is familiar with the basics of this particular branch of process algebra. Nevertheless, our approach is independent of the chosen framework as long as it supports reasoning at a trace level.

Fix a set $\mathcal{R}$ of routers, a set $\mathcal{U}$ of user, a set $\mathcal{M}$ of messages and a set $\mathcal{K}$ of keys. The set $Path$ of paths is defined by $Path = \mathcal{R}^*$. We use r to range over $\mathcal{R}$,

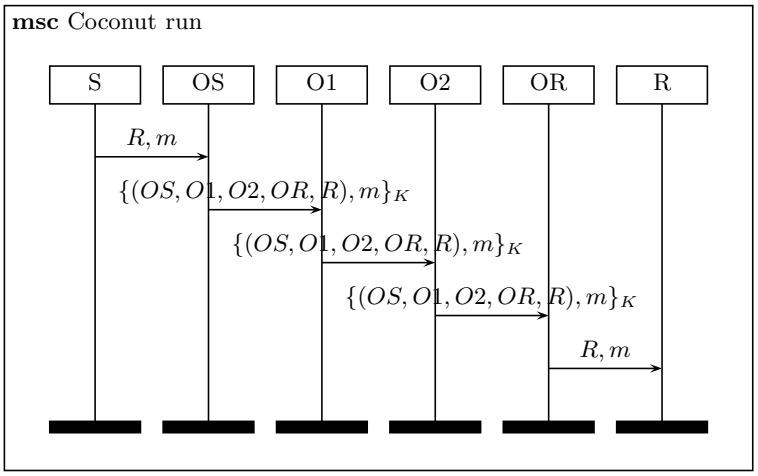

Fig. 2. Sample run of the coconut routing protocol.

u and v to range over $\mathcal{U}$, and m and p to range over $\mathcal{M}$ and Path, respectively. Fix a router assignment $\rho\colon \mathcal{U} \to \mathcal{R}$ that associates a router $\rho(u)$ with each user. We use $\mathit{site}(r)$ and $\mathit{site}(u)$ to denote the set of users u' such that $\rho(u') = r$ and $\rho(u') = \rho(u)$, respectively. We assume to be given a mapping $\mathit{pk}\colon \mathcal{R} \to \mathcal{K}$ to retrieve a public key $\mathit{pk}(r)$ of a router r. Furthermore, the topology of the router network is reflected by an undirected connected graph $\mathcal{N}$ having the set $\mathcal{R}$ as its nodes. The set $\mathcal{N}(r)$, the neighborhood of the router r, consists of all routers r' for which an edge connecting r and r' exists in the graph $\mathcal{N}$.

We use the notation $\mathit{path}(r, q)$ for the collection of those non-empty paths $r_1 \cdot r_2 \cdots r_n$ such that $\mathcal{N}(r, r_1)$, $\mathcal{N}(r_i, r_{i+1})$, for $1 \leq i < n$, and $r_n = q$. Onions are the basic objects that will be passed around in the onion routing protocol below. The set of onions $\mathcal{O}$, ranged over by o, is inductively given by

$$v \in \mathcal{U} \wedge m \in \mathcal{M} \implies \langle v, m \rangle \in \mathcal{O} \tag{1}$$

$$o \in \mathcal{O} \wedge r \in \mathcal{R} \implies \langle r, \{o\}_{\mathit{pk}(r)} \rangle \in \mathcal{O}. \tag{2}$$

The collection $\mathcal{O}_\perp = \mathcal{O} \cup \{\perp\}$ extends $\mathcal{O}$ with the dummy onion $\perp$. For a path p and onion o, the function $\mathit{pack}\colon \mathit{Path} \times \mathcal{O} \to \mathcal{O}$ is given by

$$\mathit{pack}(\varepsilon, o) = o \tag{3}$$

$$\mathit{pack}(r \cdot p, o) = \langle r, \{\mathit{pack}(p, o)\}_{\mathit{pk}(r)} \rangle. \tag{4}$$

The function pack wraps the onion o with the public keys of the routers along the path p. In particular, for a path $p = r_1 \cdot r_2 \cdots r_n$, user v and plaintext m, we have $\mathit{pack}(p, \langle v, m \rangle) = \langle r_1, \{\langle r_2, \{\ldots \langle r_n, \{\langle v, m \rangle\}_{\mathit{pk}(r_n)} \rangle \ldots\}_{\mathit{pk}(r_2)} \rangle\}_{\mathit{pk}(r_1)} \rangle$. Conversely,

the function $peel\colon \mathcal{O} \to \mathcal{O}_\perp$ is the inverse of packing, i.e., $peel(o) = o'$ if $o = \langle r, \{o'\}_{pk(r)} \rangle$ for some router r and delivers $\perp$ otherwise.

Node-oriented system description The basic building blocks in an onion routing system are the routers. We consider the process $Node_r$, with the subscript r denoting the particular router, to come equipped with a buffer B that contains the onions that are still to be delivered. In general, $B \in \mathcal{M}ul(\mathcal{O})$ is a multiset of onions. Possible actions for $Node_r$ are taken from the alphabet

$$\mathcal{A}_r = \{\, input(u, v, m), read(r', r, o), send(r, r'', o), output(v, m) \\ \mid u \in site(r), v \in \mathcal{U}, m \in \mathcal{M}, \mathcal{N}(r', r), \mathcal{N}(r, r''), o \in \mathcal{O} \,\}.$$

We fix the set of actions $\mathcal{A}$ to $\mathcal{A} = \bigcup\{\, \mathcal{A}_r \mid r \in \mathcal{R} \,\}$. A router r with buffer B can either

- input, from one of its users u, a new message m with destination v that can subsequently be forwarded along a path p from r to the router of v,
- store an onion o that is read from one of its neighboring router r' after peeling it off,
- take an onion $\langle r'', \{o'\}_{pk(r)} \rangle$ from the buffer for sending to another router r'', or
- deliver an onion $\langle v, m \rangle$ in the buffer to the user v.

Using the operators $+$ and $\sum$ to denote choice and $\cdot$ for sequential composition, we obtain the following recursive definition of the behaviour of a node:

$$\begin{aligned} &Node_r(B) \;= \\ &\quad \textstyle\sum_{u \in site(r), v \in \mathcal{U}, m \in \mathcal{M}} input(u, v, m) \cdot \sum_{p \in path(r, \rho(v))} Node_r(B \cup \{pack(p, v, m)\}) \\ &+ \textstyle\sum_{\mathcal{N}(r', r), o \in \mathcal{O}} read(r', r, o) \cdot Node_r(B \cup \{peel(o)\}) \\ &+ \textstyle\sum_{o = \langle r'', \{o''\}_{pk(r'')} \rangle \in B} send_{r, r''}(o) \cdot Node_r(B \setminus \{o\}) \\ &+ \textstyle\sum_{\langle v, m \rangle \in B} output(v, m) \cdot Node_r(B \setminus \{\langle v, m \rangle\}). \end{aligned}$$

The communication function matches read and sent events, i.e.

$$read(r, r', o) \mid send(r, r', o) = comm(r, r', o)$$

for any two routers r, r' and onion o. The set H of encapsulated or forbidden actions is given by

$$H = \{\, read(r, r', o), send(r, r', o) \mid r, r' \in \mathcal{R}, o \in \mathcal{O} \,\}.$$

Finally, using ∂_H to encapsulate partial communications and $\|$ to denote parallel composition, the onion routing network ORN is defined by

$$ORN = (\partial_H(\|_{r \in \mathcal{R}}\ Node_r(\emptyset))).$$

Thus, the onion routing network consist of a number of routers, each with a local buffer. The system starts with all routers having an empty buffer, and evolves by routers getting from and delivering to their clients and exchanging onions with other routers. Because of the choice of the communication function and encapsulation, the system ORN does not exhibit unmatched *read* and *send* actions, but *input*, *output* and *comm* actions only.

A technical issue concerns the synchronization of read and send actions. In general, the environment can influence a non-deterministic choice over the index set $path(r, \rho(v))$ for a user v and message m, by offering only a selection of reads that can match the send action that executes to the sending of $pack(p, v, m)$ to the first router along the path. This way an intruder could get control over the choice of the path connecting r and v (and, e.g., direct it via some compromised router). To prevent this, the usual trick is to insert a so-called silent action, *skip* say, just in front of $Node_r(B + \{pack(p, v, m)\})$ in the first summand. This would clutter up the further analysis dramatically, with a distinction between nodes that have or have not taken the *skip*-step after an input and path-selection. As we consider in this paper mainly an intruder model with eavesdropping capabilities only, we suppress this technicality in the remainder.

Path-oriented system description As alternative to the above node-oriented description of the network as a collection of nodes, one can follow an activity-driven approach. The node-oriented description is not very appealing from a global point of view. It is hard to identify the flow triggered by an intent of sending a message m from a user u to a user v over the network. Therefore, we view an onion routing network as a parallel composition of the process of the sending of a message by an initiating user, the passage of the associated onion along a certain path of routers, and the receipt of the message by the designated user. In order to capture the above intuition, we define processes $Comm(r, p, o)$, for a router r, path p and an onion o, to reflect that the onion o resides in packed form at the router r and still has to travel along the path p. Also, for usage in Section 4, we define processes $OR(u, v, m, p)$ representing the sending of message m from user u to v along the path p. Thus

$$\begin{aligned}
OR(u, v, m, p) &= input(u, v, m) \cdot Comm(r, p, \langle v, m\rangle) \cdot output(v, m) \\
Comm(r, \varepsilon, o) &= \varepsilon \qquad \text{if } o = \langle v, m\rangle \text{ and } v \in site(r) \\
Comm(r, r' \cdot p, o) &= comm(r, r', \langle r', pack(r' \cdot p, o)\rangle) \cdot Comm(r', p, o) \\
&\qquad \text{if } r' \in \mathcal{N}(r),\ Comm(r', p, o) \neq \delta \\
Comm(r, p, o) &= \delta \qquad \text{otherwise}
\end{aligned}$$

where, in the right-hand side, ε and δ are the successfully terminating process and unsuccessfully terminating or deadlocking process, respectively. The resulting system ORN' is then given by

$$ORN' = \sum_{r\in\mathcal{R}, u\in site(r), v\in\mathcal{U}, m\in\mathcal{M}, p\in path(r,\rho(v))} input(u, v, m) \cdot \big(ORN' \parallel Comm(r, p, \langle v, m\rangle) \cdot output(v, m)\big).$$

Note the recurrence of ORN' at the right-hand side. After the displayed input action, the system continues with the processing of the input in component $Comm(r, p, \langle v, m\rangle)$, but is also ready to initiate the sending of new messages.

Next, we would like to have that the node-oriented and path-oriented description of onion routing coincide. The former is closest to the informal description; for the latter it is immediate what its traces look like.

Theorem 1. *The systems ORN and ORN′ have the same traces.* □

For a proof of Theorem 1, one introduces some auxiliary concepts, viz. that of a distributed buffer state β and of a global communication state γ. Using these one shows, for some suitable relation C, that $C(\beta, \gamma)$ implies that the generalized systems $ORN(\beta)$ and $ORN'(\gamma)$ have the same traces. Since, in particular, it holds that $C(\emptyset, \emptyset)$ and $ORN = ORN(\emptyset)$, $ORN' = ORN'(\emptyset)$, the result follows. The characterization of Theorem 1 will be exploited in the next section, where we establish anonymity results for onion and coconut routing.

4 Anonymity properties of onion routing

In this section, we determine the anonymity group for senders and receivers engaged in a message exchange via onion routing based on the formal definition presented in Section 2. For the instantiation of Definition 1 we have to pick a system, a subset of observable actions, a set of compromised keys, a class of selection functions and a user attribution. The system under consideration is ORN with set of traces $Tr(ORN)$. We split the set of actions by considering inputs and outputs to be invisible and communications to be observable, thus $\mathcal{A}_{\text{obs}} = \{\, comm(r, r', o) \mid r, r' \in \mathcal{R}, o \in \mathcal{O} \,\}$. Furthermore, we fix a set $CN \subseteq \mathcal{R}$ of compromised nodes, i.e., a set of routers r of which the secret key corresponding to the public key $pk(r)$ is known to the intruder. Hence, the set K of compromised keys consists of $\{\, pk(r) \mid r \in CN \,\}$ that are no longer safe to use. The anonymity analysis below is with respect to the observational equivalence $\sim_{obs}$ induced by the observables $\mathcal{A}_{\text{obs}}$ and bad keys K.

The selection functions, that select the observable action of interest in a trace, are restricted to functions that point beyond a proper initialization prefix. More concretely, part of the anonymity results below depend on the fact that a router has both been recorded as a receiving and as a sending host in the trace, earlier than the selected subtrace. So, we want to distinguish an index N_t such that

$$\forall r \in \mathcal{R} \exists n_1, n_2 < N_t \colon t[n_1] = comm(r_1, r, o_1) \wedge t[n_2] = comm(r, r_2, o_2)$$

for some routers $r_1, r_2 \in \mathcal{R}$, $o_1, o_2 \in \mathcal{O}$. For such an index N_t to exist at all, we assume a fairness condition stating that every input and communication action have a successor action (communication or output) in the trace. In terms of the

causality relation $\prec_t$, to be defined in a minute, we require, for a trace t of ORN, to hold that

$$\forall i \in \mathbb{N}\colon t[i] \neq \mathit{output}(\cdot,\cdot) \rightarrow \exists j \in \mathbb{N}\colon i \prec_t j.$$

The requirement is not only technically convenient, but, more importantly, it is plausible as well. For it is realistic to postulate, that the intruder can not oversee a whole infinite trace, but only a finite part of it. Therefore, it is safe to start from the assumption that finite subprocesses will terminate within an infinite trace.

The alternative characterization of ORN captured by Theorem 1 states that a trace t of ORN is an interleaving of subtraces of the form $OR(u,v,m,p)$ for users u and v, message m and path $p \in \mathit{path}(\rho(u),\rho(v))$. In general, this does not provide a unique decomposition of the trace t. There are multiple ways to merge the finite subtraces $OR(u,v,m,p)$ into an infinite trace t. Even more so, if, e.g. due to retransmission, actions can have several occurrences in a trace. For our purposes it suffices to choose, for every position n of t, a particular subtrace $w = OR(u,v,m,p)$ such that $n \in \mathit{dom}(w)$ (exploiting the partial function interpretation of traces). More precisely, define for a trace t the relation $\prec_t$ on $\mathbb{N}$ by

$$\begin{aligned} n \prec_t m \iff\ & t[n] = \mathit{input}(u,v,m),\ t[m] = \mathit{Comm}(r,p,\langle v,m\rangle)[1], \\ & u,v \in \mathcal{U}, m \in \mathcal{M}, r = \rho(u), q = \rho(v), p \in \mathit{path}(r,q), \text{ or} \\ & t[n] = \mathit{Comm}(r,p,o)[i],\ t[m] = \mathit{Comm}(r,p,o)[i+1], \\ & r \in \mathcal{R}, p \in \mathit{path}(r), o \in \mathcal{O}, i \in \mathbb{N}, \text{ or} \\ & t[n] = \mathit{Comm}(r,p,\langle v,m\rangle)[k],\ t[m] = \mathit{output}(v,m), \\ & r \in \mathcal{R}, v \in \mathcal{U}, m \in \mathcal{M}, k = \mathit{len}(\mathit{Comm}(r,p,\langle v,m\rangle)), \\ & q = \rho(v), p \in \mathit{path}(r,q) \end{aligned}$$

such that $\forall \ell, n < \ell < m\colon t[\ell] \neq t[m]$. Then $w = OR(u,v,m,p)$, with finite domain $\mathit{dom}(w) = \{\, i_0, i_1, \ldots, i_k, i_{k+1} \,\}$, is the subtrace of t for position n if $n \in \mathit{dom}(w)$, $i_0 \prec_t i_1 \prec_t \cdots \prec_t i_k \prec_t i_{k+1}$, $[i_0] = \mathit{input}(u,v,m)$, $t[i_1,\ldots,i_k] = \mathit{Comm}(r,p,\langle v,m\rangle)$, and $t[i_{k+1}] = \mathit{output}(v,m)$.

We define, for a selection function σ, the sender attribution function α^s_σ and receiver attribution function α^r_σ, in full $\alpha^s_\sigma, \alpha^r_\sigma\colon \mathit{Tr}(ORN) \rightarrow \mathcal{U}$ by $\alpha^s_\sigma(t) = u$ and $\alpha^r_\sigma(t) = v$ if $OR(u,v,m,p)$ is the subtrace of t for position $\sigma(t)$.

Having observational equivalence and attribution in place, we continue with discussing two properties of onion routing that will be used in the proofs of anonymity results below. The first property states how a sequence of communications can be cut in two. See Figure 5.

Lemma 1 (path decomposition). *Let $r, r' \in \mathcal{R}$, $p \in \mathit{path}(r,r')$. Let q be a router on p such that $p = p_1 \cdot q \cdot p_2$ for suitable path p_1 and p_2. Then it holds that $\mathit{Comm}(s,p,o) = \mathit{Comm}(s, p_1 \cdot q, \mathit{pack}(p_2,o)) \cdot \mathit{Comm}(q,p_2,o)$.* □

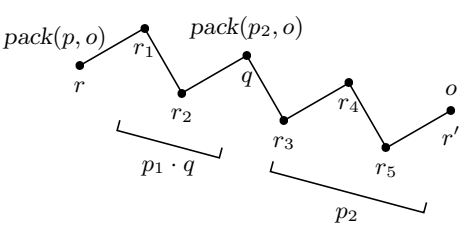

Fig. 3. path decomposition

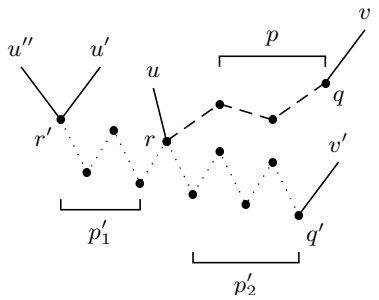

Fig. 4. cross over

The second property that we will exploit in the analysis below, states that if an outgoing communication $comm(r, r', o)$ from a router r does not originate from an input of a user of r, then there must be an earlier incoming communication $comm(r'', r, o')$ such that the outgoing onion o is obtained from the incoming onion o' by peeling off one skin. See Figure 4.

Lemma 2 (cross over). *Let $r \in \mathcal{R}$, σ a selection function and t a trace such that $\sigma(t) = i$, $t[i] = comm(r, r', o)$.*

(a) If $\alpha^s_\sigma(t) \notin site(r)$, then $j \prec_t i$, $t[j] = comm(r'', r, o')$ and $o' = pack(r', o)$ for some $j < i$, r'' and o'.
(b) If $\alpha^r_\sigma(t) \notin site(r')$, then $i \prec_t k$, $t[k] = comm(r', r'', o')$ and $o = pack(r', o')$ for some $i < k$, r'' and o'. □

We first consider the case of anonymity for senders. We distinguish between the situation where the key of router of the site is or is not compromised. As a consequence of Theorem 2 we have that, no matter how many keys have been leaked, the anonymity of a user is save as long as the key of its router is.

Theorem 2. *For a set of compromised nodes CN, onion routing has the following anonymity groups for senders: $AG_s(u) = U$ if $\rho(u) \notin CN$, and $AG_s(u) = site(u)$ otherwise.* □

The proof of the theorem exploits Lemma 1, in case $\rho(u) \notin CN$, to construct, for any trace t of ORN with $\alpha^s_\sigma(t) = u$, an alternative trace t' of ORN such that $\alpha^s_\sigma(t') = u'$. If $\rho(u) \in CN$, then, we construct a particular trace t with $\alpha^s_\sigma(t) = u$ and use Lemma 2 to rule out that any observational equivalent trace t' can be attributed to a user u' not in $site(u)$.

Next, we turn to the receiver. In the proof of Theorem 2 we exploited the initialization condition which helps in prepending subbehaviour to the subtrace under

consideration. For the case of the receiver we call upon the fairness assumption in order to find subbehaviour that extends the particular subtrace.

Theorem 3. *For a set of compromised nodes CN, onion routing has the following anonymity groups for receivers:* $AG_r(v) = \mathcal{U}$ *if* $router(v) \notin CN$, *and* $AG_r(v) = \{v\}$ *otherwise.* □

The proof of Theorem 3 is in the same vein as that of Theorem 2.

For comparison, we next consider the coconut case. Recall that in our model for coconut routing, packets are decrypted but not encrypted again at the node. Instead, the original encrypted packet is forwarded.

Theorem 4. *Coconut routing has for senders the anonymity groups* $AG_s(u) = site(u)$, *and, for receivers the anonymity groups* $AG_r(v) = site(v)$ *if the key* k *is not compromised, but* $AG_r(v) = \{v\}$ *otherwise.* □

Theorem 4, which can be proven along the same line as its onion routing counterparts, shows the weakness of our artificial coconut routing scheme. However, if incoming messages are decrypted and encrypted again using a randomized symmetric encryption schema the above reasoning does not apply. Then, the difference of onion routing vs. coconut routing lies in the robustness. If the single symmetric key is leaked, coconut routing breaks down, whereas for onion routing users at uncompromised sites remain anonymous.

The above analysis establishes the anonymity groups given the choice of parameters to the problem. A number of variations have been considered by way of experiments for our definition of anonymity groups. For example, instead of restricting the selection functions to the class Σ one can allow arbitrary selections, but demanding that each router sends itself a fake message over a random path. This does not affect the anonymity results. However, the sender anonymity drops to an isolated site for senders, but not for receivers if such a randomized initialization phase does not take place and general selection functions are allowed. Another line of variation is in the fairness assumptions or in the definition itself. A concrete alternative is to consider 'windows of observation', leading to a set-up that is simpler than the one with selection functions, but unintuitive results (as one can claim behavior just outside the window of the intruder). In fact, one could argue that the selection functions form a generalization of considering finite prefixes. A protocol, that we baptized kiwi-routing employs symmetric keys that are shared among pairs of routers. This protocol is weaker than coconut routing (whence the naming) in the sense that it breaks down as soon as one of the many shared keys gets compromised.

5 Conclusion

The achievements of our research are twofold. First of all, we have given a general and formal definition of anonymity in a trace model. Main parameters of this

definition are the attribution function which assigns to each trace a user, and the capabilities of the intruder. This allows us to calculate a user's anonymity group, i.e. the collection of other users that cannot be distinguished from this user by the intruder. Our definition is of a qualitative nature and discards quantitative aspects. This means that we only consider statements such as "could this behaviour be attributed to some other user", rather than "what is the chance that this behaviour is caused by some other user". It is our believe that a formal quantitative analysis of a security protocol can only be achieved after first having developed a proper qualitative analysis methodology. In future research we wish to investigate the use of tool support to analyze privacy protocols and to adapt our approach to a quantitative setting.

The second result of our research is the formalization of a basic onion routing protocol and its analysis. By abstracting from several details we were able to concentrate on what we consider the protocol proper and to formalize some of the insights expressed by the designers of the protocol. By varying over the user attribution function we could analyze the anonymity of both the receiver and sender of some intercepted message. Reasoning about different attribution functions, such as "did u ever send a message", follows the same line. Due to the restrictions on the intruder's choice function, we were able to express conditions on the initialization of the protocol that guarantee full privacy. It is not the case that during this initialization phase senders of messages can be traced back. The size of a user's anonymity group expands during this phase until it comprises all other users. It would be interesting to express the anonymity group of each user during initialization as a closed expression.

One of our aims was to understand why the onion routing protocol has its current shape and under which conditions its privacy properties are satisfied. Thereto we compared it to several weaker protocols, of which we have discussed the coconut routing protocol here only. This comparison explains what the implications are of simplifying the layered messages. Coconut routing hardly guarantees privacy. While performing our analysis, it turned out quite naturally that the onion routing protocol requires randomized encryption to guarantee full privacy. Without such randomization the protocol is vulnerable to guessing attacks, when the intruder seeks to relate incoming and encryptions of outgoing traffic.

It would be interesting to analyze more complex versions of the onion routing protocol, such as an extension of the protocol with connections, like in the original onion routing protocol. Further validation of our methodology would not only require to consider other protocols, but also stronger intruder models. In the case of the onion routing protocol it is conjectured that an active intruder could not threaten privacy more than a passive (eavesdropping) intruder. Since denial-of-service attacks is a topic of research of its own, we will not consider these attacks in the course of our privacy research.

A final topic of future research is the formalization of other privacy notions, such as unlinkability, pseudonymity and unobservability. Initial research indicates that their formalization follows the same line as the formalization of anonymity.

References

1. J.C.M. Baeten and W.P. Weijland. *Process Algebra*, volume 18 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 1990.
2. J.A. Bergstra, A. Ponse, and S. A. Smolka. *Handbook of Process Algebra*. Elsevier, 2001.
3. R. Clarke. Introduction to dataveillance and information privacy, and definitions of terms. `http://www.anu.edu.au/people/Roger.Clarke/DV/Intro.html`, 1999.
4. C.J.F. Cremers, S. Mauw, and E.P. de Vink. Defining authentication in a trace model. In T. Dimitrakos and F. Martinelli, editors, *Proc. FAST 2003*, 1st International Workshop on Formal Aspects in Security and Trust, pages 131–145, Pisa, 2003. IITT-CNR technical report.
5. C. Díaz, J. Claessens, S. Seys, and B. Preneel. Information theory and anonymity. In B. Macq and J.-J. Quisquater, editors, *Proc. 23rd Symposium on Information Theory in the Benelux*, pages 179–186. Université Catholique de Louvain, 2002.
6. EPIC. Comments of the Electronic Privacy Information Center. `www.epic.org/privacy/drm/`, 2002.
7. W.J. Fokkink. *Introduction to Process Algebra*. Texts in Theoretical Computer Science, an EATCS Series. Springer, 2000.
8. Association for Automatic Identification and Mobility. Rfid.org. `www.aimglobal.org/technologies/rfid/`, 2004.
9. D.M. Goldschlag, M.G. Reed, and P.F. Syverson. Hiding routing information. In R.J. Anderson, editor, *Proc. 1st International Workshop on Information Hiding*, pages 137–150, Cambridge, 1996. LNCS 1174.
10. ISO. Common Criteria–ISO/IEC/15408. `http://csrc.nist.gov/cc/`, 1999.
11. M. Korkea-aho. Anonymity and privacy in the electronic world, 1999. Seminar on Network Security, Helsinki University of Technology.
12. A. Pfitzmann and M. Köhntopp. Anonymity, unobservability, and pseudonymity. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, pages 1–9. LNCS 2009, 2001.
13. M.K. Reiter. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, pages 66 – 92, 1998.
14. S. Schneider and A. Sidiroupoulos. CSP and anonymity. In *Proc. ESORICS'96*, pages 198–218. LNCS 1146, 1996.
15. A. Serjantov and G. Danezis. Towards an information theoretic metric for anonymity. In H. Federrath, editor, *Proc. PET 2002*, pages 41–53. LNCS 2482, 2003.
16. P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an analysis of onion routing security. In H. Federrath, editor, *Designing Privacy Enhancing Technologies*, pages 96–114. LNCS 2009, 2001.
17. P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and onion routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 1997.