



Article

# Achieving Sender Anonymity in Tor against the Global Passive Adversary

Francesco Buccafurri <sup>\*,†</sup>, Vincenzo De Angelis <sup>†</sup>, Maria Francesca Idone <sup>†</sup>, Cecilia Labrini <sup>†</sup> and Sara Lazzaro <sup>†</sup>

Department DIIES, University of Reggio Calabria, Via Università 25, 89122 Reggio Calabria, Italy; vincenzo.deangelis@unirc.it (V.D.A.); mariafrancesca.idone@unirc.it (M.F.I.); cecilia.labrini@unirc.it (C.L.); saralazzaro20@gmail.com (S.L.)

\* Correspondence: bucca@unirc.it

† These authors contributed equally to this work.

**Abstract:** Tor is the de facto standard used for anonymous communication over the Internet. Despite its wide usage, Tor does not guarantee sender anonymity, even in a threat model in which the attacker passively observes the traffic at the first Tor router. In a more severe threat model, in which the adversary can perform traffic analysis on the first and last Tor routers, relationship anonymity is also broken. In this paper, we propose a new protocol extending Tor to achieve sender anonymity (and then relationship anonymity) in the most severe threat model, allowing a global passive adversary to monitor all of the traffic in the network. We compare our proposal with Tor through the lens of security in an incremental threat model. The experimental validation shows that the price we have to pay in terms of network performance is tolerable.

**Keywords:** anonymous communication systems; Tor; Onion; censorship resistance



**Citation:** Buccafurri, F.; De Angelis, V.; Idone, M.F.; Labrini, C.; Lazzaro, S. Achieving Sender Anonymity in Tor against the Global Passive Adversary. *Appl. Sci.* **2022**, *12*, 137. <https://doi.org/10.3390/app12010137>

Academic Editors: David Megías and Eui-Nam Huh

Received: 14 October 2021

Accepted: 17 December 2021

Published: 23 December 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

The Tor overlay network [1] is the most popular anonymous communication protocol used for low-latency network applications. Tor is based on the Onion protocol [2]. This protocol is based on two concepts: relay nodes (also called *Tor routers*) and layered encryption. Relay nodes act as proxies in an Onion route. Each relay node receives its message from the preceding one and forwards it to the next, until the destination is reached. Differently from random walk [3], the route is deterministic and chosen by the sender. Moreover, the message is wrapped through layered encryption, which the sender can apply by knowing the cryptographic keys of all the relay nodes of the route. This way, each node is able to drop an encryption layer, and can see the address of the next relay node to which the still encrypted message should be forwarded. Eventually, the message with only one layer of encryption reaches the destination. According to this scheme, each node in the route only knows the address of the preceding node and the address of the next node. Therefore, by design, the first relay node knows the address of the sender. *Sender anonymity* is then not supported if we allow the adversary to control the first relay node. The practical impact of this weakness is that sole collaboration with an Internet service provider allows the adversary to detect that a user is utilizing the Tor system. Sender anonymity is obviously broken in a severe threat model with a global passive adversary, able to monitor all the traffic in the network. Anyway, breaking sender anonymity is not enough to nullify the final goal of the protocol, which is *relationship anonymity*. Indeed, the aim of Tor, as in general happens for an anonymous communication network, is to prevent the adversary from detecting that a given sender is communicating with a given recipient. Consider that, despite the fact that anonymity services are often used for criminal purposes, there are a lot of ethical applications of anonymous routing, including censorship resistance. However, relationship anonymity can be broken in Tor in a global passive adversary model.

As a matter of fact, Tor is vulnerable to many passive attacks [4,5], allowing traffic de-anonymization. It can be easily recognized that if the adversary can monitor the traffic at the bounds of the Tor circuit (i.e., the first and the last router), traffic analysis attacks break relationship anonymity [6,7], thereby fully de-anonymizing the communication.

In this paper, our aim is to overcome the above drawbacks of Tor, by achieving sender anonymity (in the sense of *communication k-anonymity* [8]) in the most severe threat model, in which a global passive adversary is allowed, which monitors all the traffic in the network. Recall that sender anonymity is enough to guarantee relationship anonymity, as stated in [7]. Therefore, we obtain effective protection of users' privacy.

The approach we use to obtain sender anonymity in Tor is to hide the sender within an anonymity set of nodes built as a *ring* of potential senders. To prevent the adversary from detecting the initiator of the communication, we equip the ring with cover traffic that the senders can opportunistically use to send their messages, by filling one or more of the circulating tokens. Thanks to probabilistic encryption, empty and filled tokens are indistinguishable for the adversary. The route Tor is then built from a proxy node of the ring to the destination. The adversary can see that a node of the ring is working as a proxy node, but it is not able to understand which node the sender is among the nodes of the ring. Traffic analysis attacks are not possible due to the cover-traffic mechanism.

Our approach can be related to *buses*[9,10], as we also consider a pre-determined route that is opportunistically used by the sender. However, there is a crucial difference. In buses, the fixed route is a Eulerian path passing through all the nodes, including thus all the possible pairs of sender–receiver. This is an impractical solution resulting in intolerable communication latency in a large network (such as the Internet).

Instead, our approach allows us to modulate the size of the anonymity set to a value that fulfills reasonable anonymity requirements, without introducing intolerable latency times, and importantly, relying on the existing Tor system. The feasibility of our approach was tested through careful experimental analysis conducted by simulations. Therefore, this research involved both theoretical and experimental analysis.

To the best of our knowledge, there is no proposal in the literature aimed at equipping the Tor protocol with sender anonymity against a global passive adversary. A proposal with some similarities to our paper (as both papers take inspiration from the original idea of buses [10], as discussed above) is given in [11]. However, their method [11], as clearly stated in the paper, does not provide protection against a global passive adversary, because the observation of the initiator of the ring construction breaks sender anonymity. On the contrary, sender anonymity against a global passive adversary in all the (even preliminary) phases of the communication is the objective achieved by our approach, which purposefully advances the state of the art. Moreover, our paper treats and solves the problem for the case of complete bi-directional communication, i.e., a request from the sender to the recipient and a response from the recipient to the sender. Observe that, in general, this is not trivial when anonymity should be maintained. Indeed, the response cannot be simply implemented as a different forward tunnel directed from the recipient to the sender; otherwise, simple intersection attacks would break anonymity. The method in [11] does not facilitate a response. This makes it applicable only for unidirectional communications, which is a very strong limitation.

The structure of the paper is the following. In Section 2, we review the related literature. We provide some basic notions needed to understand the proposed protocol in Section 3. The protocol is presented in Section 4. The introduction of a certain degree of fault tolerance is described in Section 5. The computational complexity of our solution is discussed in Section 6. The security of the protocol is analyzed in Section 7. In Section 8, we report the results of an experimental validation of our approach. Finally, in Section 9, we draw our conclusions.

## 2. Related Work

The issues most relevant to this paper are the vulnerabilities which Tor suffers from.

As stated by the creators themselves [1], the Tor overlay network, based on Onion routing [2], does not provide anonymous guarantees in the severe threat model of a global passive adversary [4], which is able to observe the entire traffic of the network.

Anyway, even if we relax the powers of the adversary, many attacks are still effective [5,12,13]. The most famous class of attacks is represented by the traffic analysis attacks [14–16] in which the adversary analyzes the traffic to find correlations. Among the traffic analysis attacks there are the timing attacks [17–19], in which the adversary observes the timing of the messages arriving at and leaving from the nodes to find correlations. Other interesting subclasses of traffic analysis attacks are traffic confirmation attacks [20], in which the adversary controls and observes two possible end-relays of a Tor circuit to conclude that they really belong to the same circuit, and watermarking attacks [21], in which the adversary manipulates the traffic stream by introducing an identifiable pattern. Another category of attacks target the router selection used to build the Tor circuit. Indeed, the standard selection is based on network and CPU performance reported by the nodes themselves. This enables self-promotion attacks [22]. A countermeasure can be found in [23].

The performance of Tor was investigated in [24–26]. Performance analysis in relation to de-anonymization attacks was performed in [27].

In our approach, we extend Tor achieving sender anonymity (and then relationship anonymity) [7] in the sense of *communication k-anonymity* [8], against a global passive adversary. This goal can be reached only with the introduction of cover traffic [28] (as required by our approach).

Among the approaches supporting cover traffic, the most significant are *mixnets*, originally proposed in [29], and *buses* [9,10,30].

In the literature, several proposals include cover traffic in mixnets [31–34]. The introduction of cover traffic makes traffic analysis more difficult. For example, a possible approach is to introduce cover traffic to maintain a constant transmission rate. A very recent mixnet-based approach designed for the network layer was presented in [35]. However, it does not provide sender anonymity against a global adversary. Another relevant approach in this category, even if dated (but still very solid), is Tarzan [32]. As discussed in [36], mixnets, in general, require a suitable amount of cover traffic.

More related to our work are buses, as we also consider a pre-determined route that is used by the sender. However, buses are unrealistic in a large network (such as the Internet), since the fixed route is a Eulerian path passing through all the nodes, including thus all the possible pairs of sender–receiver.

Similar considerations can be made for DC-Nets [37], based on a secure multi-party cryptographic protocol, in which it is required that all participants are involved in every run of the protocol and initially share a pairwise key.

This paper considerably extends a work-in-progress paper [38]. Reference [38] just presented the rough idea underlying the protocol. Specifically, in that paper, the approach is only sketched out, and it refers to the original Onion approach, with no fault tolerance, no real-life contextualization in the Tor system. Moreover, no detailed security and complexity analyses were performed, and no experimental evaluation was included.

### 3. Background and Notation

#### 3.1. Anonymity

We recall some background notion taken from [7]. An anonymous communication network may offer:

1. *Sender anonymity*: the adversary cannot sufficiently identify the sender in a set of potential senders, called the *sender anonymity set*;
2. *Recipient anonymity*: the adversary cannot sufficiently identify the recipient in a set of potential recipients, called the *recipient anonymity set*;

3. *Relationship anonymity*: the adversary cannot sufficiently identify that a sender (in a set of potential senders) and a recipient (in a set of potential recipients) are communicating. According to [7], sender anonymity implies relationship anonymity.

Observe that the definition of anonymity given in [7], with the use of the term sufficiently, means “both that there is a possibility to quantify anonymity and that for some applications, there might be a need to define a threshold where anonymity begins”.

### 3.2. The Tor Network

The Tor network is an overlay network, based on TCP/TLS connections, consisting of multiple relay routers called *Onion routers* (OR). Each client runs locally an *Onion proxy* (OP) which establishes a virtual circuit of ORs to communicate anonymously with the destination. To build a circuit, the OP contacts periodically a trusted server called *Directory Server* (DS) that keeps information about the state of the network and provides the OP with *router descriptors* of the ORs. These router descriptors contain the IP addresses and the public keys of the ORs, along with their network information, such as the bandwidth. Then, the OP selects, according to some strategies, a number  $n$  of OR relays that form the virtual circuit. By default,  $n = 3$ . The first OR is called the *entry router*, the second the *middle router* and the last the *exit router*. Once the three ORs have been selected, the OP starts a set-up phase to build the virtual circuit. This phase is performed in such a way that each OR only knows the previous and the next node of the path. Moreover, in this phase, the OP exchanges some messages with the ORs, which include some Diffie–Hellman (DH) parameters, to share a secret key. These messages are encapsulated into *control cells* of a fixed size of 512 bytes. Since the OP has to be sure about the authenticity of the ORs, the DH parameters are encrypted by using the public keys of the ORs. At the end of this set-up phase, the OP shares a secret key with each OR. These keys are used by the OP to encrypt (symmetrically) in Onion fashion the messages intended for the destination. Once the circuit is established, the OP sends the messages to the destination encapsulated into *relay cells* of size of 512 bytes. These relay cells include a header of 3 bytes in plaintext plus 11 bytes encrypted for the exit router. Therefore, the effective payload is 498 bytes.

Through this paper, for both symmetric and public-key encryption, we denote by  $E_k(M)$  the encryption of a message  $M$  with key  $k$ . Similarly, we denote by  $D_k(C)$  the decryption of the ciphertext  $C$  with (symmetric or public) key  $k$ . Even though we do not explicitly highlight this aspect, the encryption we consider in this paper is only probabilistic, in such a way that, for an eavesdropper, two different encryptions of the same message are unlinkable.

## 4. The Proposed Protocol

In this section, we describe our protocol, called Ring2Tor, which achieves sender anonymity even in the most severe threat model including a global passive adversary. We denote by (*client*) *nodes* the nodes that collaborate in the protocol without playing the role of Tor routers. Senders are among the client nodes. Moreover, we have in the network  $n_d$  destination hosts, which are distinct from client nodes and Tor routers.

The description of the protocol is given in three main steps. The first step is describing the ring manager and the token-based mechanism. Some management functions are illustrated, along with the basic mechanism for implementing anonymity for the sender. The second step is describing the *set-up phase*. This is the phase in which keys are exchanged, the setting of further parameters is executed and cover traffic is established. This is a preliminary step to make possible the anonymous communication, which is explained in the last step of the description, denoted as *communication phase*.

### 4.1. Ring Manager and Token-Based Mechanism

In this section, we describe the basic mechanism of our approach that allows us to provide the sender with anonymity against a global passive adversary.

We assume the presence of a *ring manager* (RM) that partitions the nodes of the network in several *rings*.

The ring manager selects the nodes forming a ring in such a way that the background knowledge does not allow a possible adversary to have more information than the uniform distribution of senders. In other words, given a ring, any node of the ring is potentially a sender (with no probability bias). This is achieved by selecting, for a given ring, hosts belonging to the same, even large, geographical region.

A ring is a sequence of  $k$  nodes such that each node has exactly a *preceding* (*prec*, for short) and a *next* node. In our setting, each node only knows its *prec* and its *next* node. Several messages, called *tokens*, move through the ring. There are two kinds of tokens. The first type is used in the set-up phase. The second type is used in the succeeding communication phase. The detail will be discussed next. Tokens are filled by senders to deliver their messages to a proxy node, which, once a Tor circuit is established, sends them to the destination host. To obtain that any eavesdropper is unable to distinguish an empty token from a filled token, each node encrypts the token with a symmetric key shared with its next node.

RM maintains, along with the next node, the public keys and the network addresses of each node of the network. For each ring, each belonging node receives from RM the set of the public keys of the other nodes of the ring, and among these keys, the information about which is the public key associated with the next node in the ring. In this paper, we assume that RM is a centralized entity.

#### 4.2. Set-Up Phase

The first purpose of this phase is to exchange a set of symmetric keys between the nodes of a ring. These keys will be used to encrypt the messages without requiring the complexity of public-key encryption.

We first introduce some notation. Given a ring, we denote by  $r_1, \dots, r_k$  the  $k$  nodes forming the ring, in order. Given a node  $r_i$ , we denote by  $next(r_i)$  the next element in the ring, that is,  $r_{(i\%k+1)}$ , where  $\%$  is the operator *mod*. We denote by  $PK_{r_i}$  the public key associated with the node  $r_i$  and by  $addr(r_i)$  its network address.

Now, we can describe how key exchange is executed. This is done in detail next. We have two kinds of key exchange. The first is aimed at providing each node with a symmetric key shared with the next node. These keys are used to implement hop-by-hop encryption when messages turn in the ring. This key exchange is called *forward key exchange*, and it is described in detail next, in Section 4.2.1.

The second kind of key exchange is aimed at obtaining key sharing between the sender and the proxy node. However, since both roles of sender and proxy can be played by all the nodes in the ring, the key exchange mechanism involves every pair of nodes. Synthetically, each node of the ring exchanges a symmetric key with the other  $k - 1$  nodes. Observe that, even though a key is exchanged between two nodes  $A$  and  $B$ , a different key will be exchanged between  $B$  and  $A$ . Indeed, the two keys will be used for different purposes depending on whether the node plays the role of *sender* or *proxy*. Therefore, the two keys are called the *sender key* and *proxy key*, respectively. A requirement of this phase is that, if  $A$  exchanges a key with  $B$ ,  $B$  learns nothing about the network address of  $A$ . The detail of this mechanism, called *sender and proxy key generation*, is provided next, in Section 4.2.3. Since the above keys will be included into special tokens, before describing the key generation mechanism, we describe, in Section 4.2.2, how such tokens are arranged.

##### 4.2.1. Forward Key Exchange

Each node  $r_i$  receives from RM the set  $Q$  of the public keys of the nodes of the ring it belongs to,  $addr(next(r_i))$ , and among  $Q$ , the information about which public key is associated with  $next(r_i)$  (the associations of the other keys with the proper network address remain unknown to  $r_i$ ). The address of the next node will be used to forward tokens.



Initially, each node  $r_i$  exchanges a symmetric key called *forward key* with its next node. This key is used only to encrypt the token hop-by-hop. In detail, as the exchange of keys occurs between the OP and the first OR in Tor (see Section 3), each node  $r_i$  generates a public DH parameter  $y_i$  and encrypts it with the public key  $PK_{next(r_i)}$ , obtaining  $C = E_{PK_{next(r_i)}}(y_i)$ .  $C$  is sent to  $next(r_i)$  (we recall that  $r_i$  knows  $add(next(r_i))$ ). The latter decrypts  $y_i$ , generates the *forward key*  $k_{r_i}$  and replies to  $r_i$  with its public DH parameter  $\bar{y}_{next(r_i)}$  along with the hashed value  $H(k_{r_i})$  (in plaintext). In summary, each node  $r_i$  shares a forward key  $k_{r_i}$  with its next node, and the tokens can be properly encrypted hop-by-hop.

#### 4.2.2. Token Generation

After exchanging the forward keys, at a given time  $t_0$ , each node  $r_i$  generates  $k - 1$  empty tokens and sends them to its next. In turn,  $next(r_i)$  forwards the tokens to its next, and so on. Each token is encrypted by  $r_i$  with  $k_{r_i}$ ; then it is sent to  $next(r_i)$ , which decrypts it with  $k_{r_i}$ , processes the token, re-encrypts it with  $k_{next(r_i)}$  and forwards it to  $next(next(r_i))$ .

The structure of these tokens is the following:  $\langle F, PDH, R, H \rangle$  where  $F$  is a flag denoting whether the token is empty ( $F = 0$ ) or filled ( $F = 1$ ),  $PDH$  is a field containing a public Diffie–Hellman parameter (possibly encrypted),  $R$  is a random playing the role of identifier and  $H$  is a hashed value (the exact meaning of these fields will be clear in the following). Observe that  $PDH$ ,  $R$  and  $H$  are meaningful only if  $F = 1$ . The tokens are born with  $F = 0$ . Therefore, at the beginning, there are  $k(k - 1)$  empty tokens turning in the ring.

Starting from a time  $t_1 > t_0$ , each node  $r_i$  waits a random time  $\delta_i$ , and then fills the first available empty token, as explained in the following.

#### 4.2.3. Sender and Proxy Key Generation

First,  $F$  is set to 1. Then,  $r_i$  selects a random public key  $PK_{r_j}$  from  $Q \setminus \{PK_{r_i}\}$ .  $r_i$  selects its public DH parameter  $y_{ij}$  and encrypts it with  $PK_{r_j}$ , thus obtaining  $C_{ij} = E_{PK_{r_j}}(y_{ij})$ . Then,  $PDH$  is set to  $C_{ij}$ .  $R$  is set to a random value used by  $r_j$  to reply with its public DH parameter, which is needed by  $r_i$ . This DH parameter is used in the construction of the key that  $r_i$  will use to send a message by using  $r_j$  as a proxy. This key  $k_{ij}$  is called the *sender key* for  $r_i$  (with respect to  $r_j$ ), and the *proxy key* for  $r_j$  (with respect to  $r_i$ ). Finally,  $H$  is filled with random bits.

The token  $T$  is encrypted by  $r_i$  with  $k_i$ , by obtaining  $C_T = E_{k_i}(T)$ . Then,  $C_T$  is sent to  $next(r_i)$ .

When  $C_T$  reaches  $next(r_i)$ , it decrypts  $C_T$ , by obtaining  $T$ , and since  $F = 1$ , it tries to read the field  $PDH = C_{ij}$  of  $T$ . If  $next(r_i) \neq r_j$ ,  $next(r_i)$  is not able to decrypt such a field, and then it re-encrypts the token with the forward key  $k_{next(r_i)}$  shared with  $next(next(r_i))$  and forwards the token. The token moves through the ring until it reaches  $r_j$ . At this point,  $r_j$  decrypts  $C_{ij}$  and obtains  $y_{ij}$ , with which it generates the key  $k_{ij}$  which is shared with  $r_i$ . The token is filled as follows.  $F$  remains set to 1.  $PDH$  is set to  $\bar{y}_{ji}$ .  $\bar{y}_{ji}$  represents the public DH parameter of  $r_j$  that will be used by  $r_i$  to generate the key  $k_{ij}$ .  $R$  remains unaltered, and finally,  $H$  is set to the hashed value  $H(k_{ij})$ . This new token moves through the ring until  $r_i$ . Observe that all the nodes between  $r_j$  and  $r_i$ , after decrypting the token with their forward keys, understand that the token is used to reply to a node, but are unaware of the sender and the recipient of this token.

When  $r_i$  receives the token, it identifies the token as a reply of  $r_j$  thanks to the random  $R$ . Then,  $r_i$  can generate the key  $k_{ij}$  as  $r_j$ . This token is then disposed by  $r_i$ . Finally,  $r_i$  drops from the set  $Q$  the node  $r_j$ . Note that any external observer only knows that a key was exchanged by a given node  $r_i$ , but does not know with which node.

The entire process (which started at time  $t_1$ ) is repeated  $k - 2$  times, until all  $k_{ij}$  are exchanged.

When all the  $k(k - 1)$  tokens are disposed of, each node  $r_i$  owns (in addition to the forward key) two symmetric keys  $k_{ij}$  and  $k_{ji}$  shared with each other node  $r_j$  of the ring. The key  $k_{ij}$  represents a sender key for  $r_i$ , since it used by  $r_i$  when has to send a message by

selecting  $r_j$  as a proxy node (see next section). On the other hand,  $k_{ij}$  represents a proxy key for  $r_j$ , since it is used by  $r_j$  when plays the role of proxy node.

In Figure 1, the sequence diagram of the set-up phase is depicted.

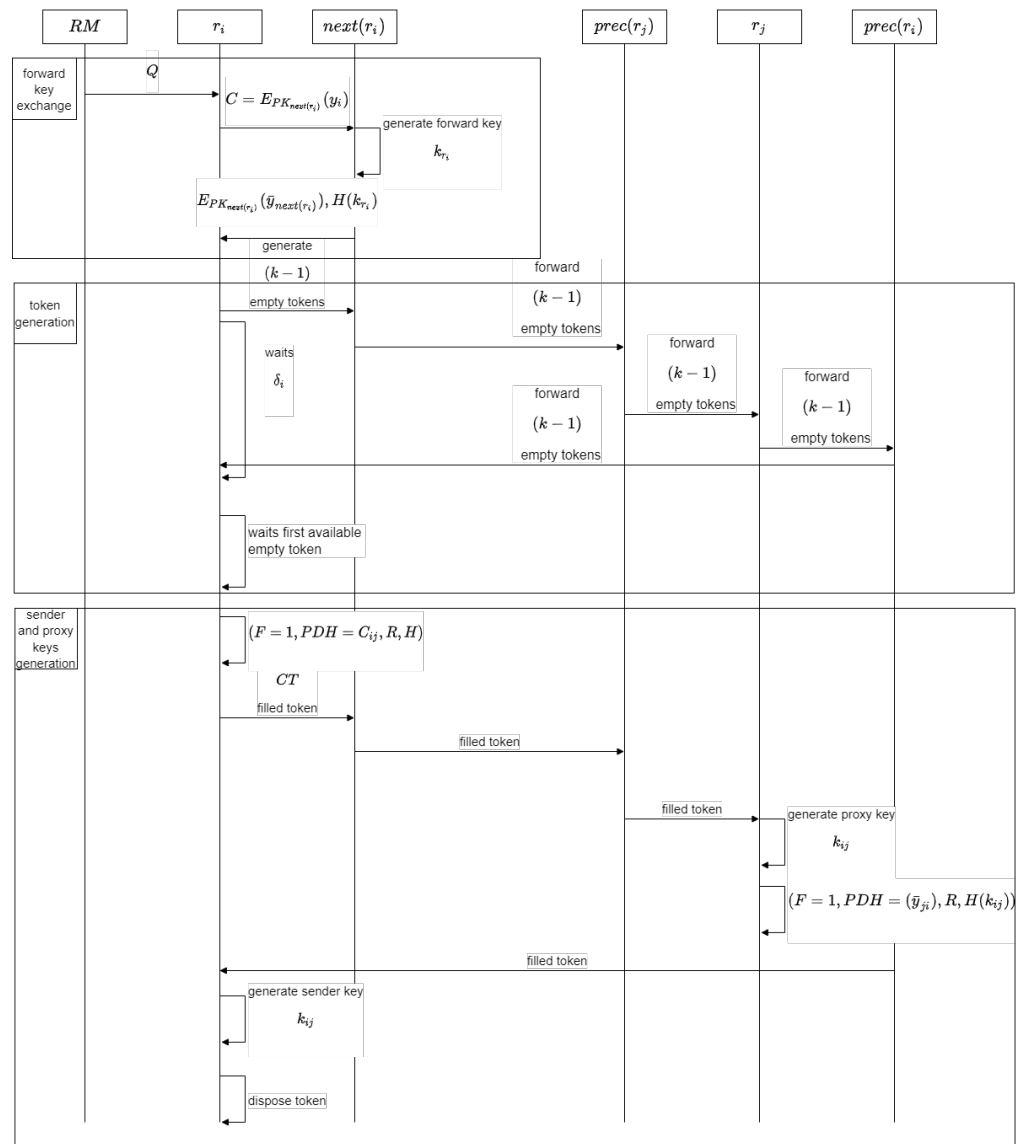


Figure 1. Set-up phase.

### 4.3. Communication Phase

In this section, we describe the core of our protocol, which is the communication between a sender and recipient. We remark that the communication is bi-directional, in the sense that we address both the request and the response. We split the description of the communication phase into three parts. The first part is the structure of tokens in which messages are encapsulated. Observe that these tokens are different from those used in the set-up phase, which we described in Section 4.3.1. After describing the structure of the tokens, we show how tokens are generated (see Section 4.3.2. Finally, in Sections 4.3.3 and 4.3.4, we describe how anonymous communication is established between a sender and a recipient.

#### 4.3.1. Structure of the Token

As in the set-up phase, in the communication phase, a token-based mechanism is enabled. We assume that a given number of tokens move through the ring encrypted, hop-by-hop, from one node to the next, with the forward key exchanged in the set-up phase.

These tokens are managed (generated and disposed) by some nodes of the ring according to the network requirements (throughput, bandwidth, etc.). The specifications of these requirements are discussed in Section 8.

The structure of a communication-phase token is the following:  $\langle F, HID, CI, DA, P \rangle$ . In Figure 2, an expanded description of this structure is reported.

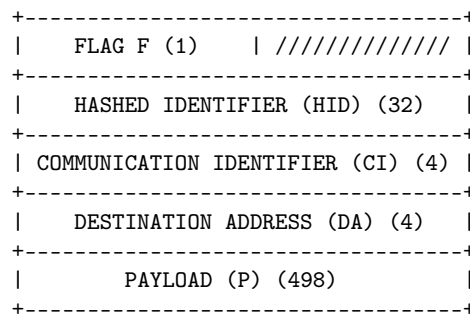


Figure 2. Structure of the token.

As the communication phase is the core of Ring2Tor, we describe in detail how the token is organized. Its size is 539 bytes, of which 41 are reserved for the header, and 498 for the payload. The size of the payload is set to the same value as the size of the payload of the relay Tor cells.

First, we describe the meaning of the field  $F$ . It is composed of two bits (even though we reserve 1 byte for this field), with following possible meanings: 00 means *empty* token; 01 means token *reserved* for a given communication identifier; and 10 means that it is *used* for a message. A token in the state 01 (reserved) or 10 (used) is said to be *filled*.

During the description of the protocol, which we provide next, the meanings of the remaining fields are clarified.

#### 4.3.2. Token Generation

Consider now the process of token generation. When a token is generated by a node  $r_g$ , the fields are set as follows.  $F$  is set to used (i.e., 10).  $r_g$  picks randomly from the set  $Q$  (where  $Q$  is the set of all the public keys of the ring) a public key, say  $PK_{r_p}$ , associated with the node  $r_p$ . The field  $HID$  is set to  $H(PK_{r_p})$ . It is used as an identifier to allow  $r_p$  to recognize that this token is intended for it. Finally, the field  $DA$  includes the encryption  $\bar{S}$  with the sender key (of  $r_g$ )  $k_{gp}$  of a fixed string  $S$  different from any other network address. This string allows  $r_p$  to identify the fact that this token, if even used, does not contain any message to forward outside the ring (see below), but it has to be emptied by  $r_p$ . The reason why the token is not directly generated empty derives from security aspects. The security analysis is provided in Section 7. The other fields ( $CI, P$ ) are filled with random bits.

The entire token is then encrypted with the forward key  $k_{r_g}$  and sent to  $next(r_g)$ . This node decrypts the token, and with the state of the token being filled, through the field  $HID$ , it checks whether this token is intended for it. In this case (i.e.,  $r_p = next(r_g)$ ), it processes the token. Otherwise, the token is encrypted, as usual, by  $next(r_g)$  with the forward key  $k_{next(r_g)}$  and sent to  $next(next(r_g))$ . The token moves through the ring until it reaches  $r_p$ .

At this point,  $r_p$  verifies that it has been selected as recipient of the token, even though it does not know that the token was generated by  $r_g$ . Therefore,  $r_p$  tries to decrypt the fields  $CI, DA, P$  with all its  $k - 1$  proxy keys until it finds the correct key  $k_{gp}$ . Since  $D_{k_{gp}}(DA) = S$ ,  $r_p$  knows that it has to empty the token. Thus,  $r_p$  sets  $F$  to 00 and  $HID = H(PK_{next(r_p)})$ . In this case, we say that  $next(r_p)$  will play the role of *proxy node* (with respect to a potential sender for a communication identifier not established yet). The other fields are set to random bits.

$r_p$  encrypts the token with the forward key  $k_{r_p}$  (shared with its next) and forwards it to  $next(r_p)$ . The empty token crosses the ring encrypted hop-by-hop, as usual.



The process of generation of the tokens is represented in the sequence diagram in Figure 3.

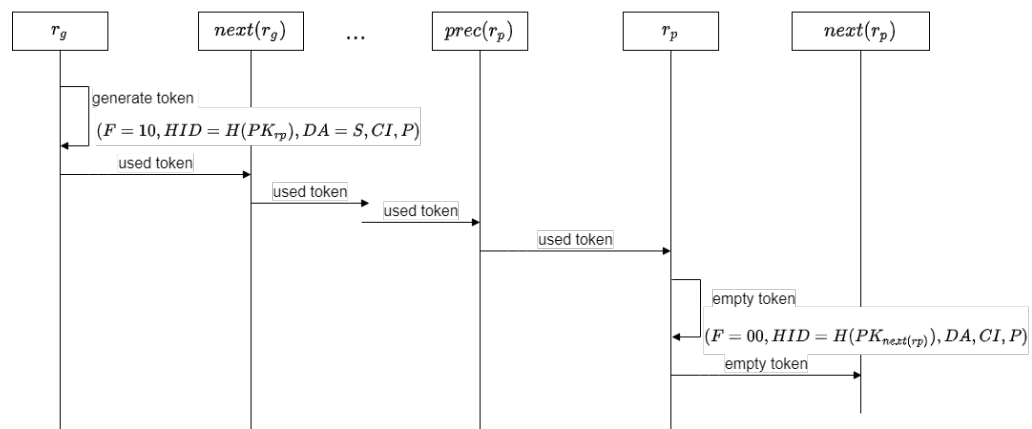


Figure 3. Process of generation of the tokens.

### 4.3.3. Transmission of a Message

Consider a node  $r_i$  that wants to send a message  $M$  to a destination  $D$  (outside the ring). Suppose  $M$  is already encrypted for  $D$ . First,  $r_i$  splits  $M$  into blocks  $M_1, \dots, M_q$  ( $q \geq 1$ ) with size 498 bytes (i.e., the size of the payload  $P$  of a token).  $r_i$  waits for the first empty token (with  $F = 00$ ). Let be  $HID = H(PK_{r_j})$  (this means that  $r_j$  will play the role of proxy node for a communication session started by  $r_i$ , as we will see next). Through  $HID$ ,  $r_i$  identifies the public key  $PK_{r_j}$  and the corresponding sender key  $k_{ij}$ .

The token is filled as follows.  $F$  is set to 10 (used).  $HID = H(PK_{r_j})$  is unaltered.  $CI$  is set to  $E_{k_{ij}}(R)$  where  $R$  is a random value identifying the current communication session associated with the sender key  $k_{ij}$  (note that for a given communication session, a Tor circuit will be established outside the ring). The field  $DA$  includes the encryption with key  $k_{ij}$  of the network address of the destination  $D$ . Observe that the size of this field is 4 bytes, and thus is compliant only with IPv4. Obviously, for IPv6, the size should be increased. Moreover, the TCP port is not included in this field for privacy reasons. It will be included in the payload encrypted at application layer. Finally,  $P$  is set to  $E_{k_{ij}}(M_1)$  (possibly padded, if  $q = 1$ ). The token moves through the ring (encrypted hop-by-hop) until it reaches  $r_j$ .

Regarding the other messages  $M_t$  (with  $2 \leq t \leq q$ ),  $r_i$  waits for either (1) an empty token with  $HID = H(PK_{r_j})$  or (2) a reserved token ( $F = 01$ ) with  $HID = H(R)$ , meaning that the token is reserved for the communication session started by  $r_i$  identified by  $R$ .

In both cases, the token is filled as follows.  $F$  is set to 10,  $HID$  is set to  $H(R)$  in case (2) (indeed, in case (1) it is already set with this value),  $CI = E_{k_{ij}}(R)$ ,  $DA$  includes the encryption with key  $k_{ij}$  of the network address of the destination  $D$  and  $P = E_{k_{ij}}(M_t)$ . Additionally, these tokens move through the ring until they reach  $r_j$ . Eventually, all the blocks of the message  $M$  reach the same proxy node  $r_j$ , which will use the same Tor circuit.

We now see how such Tor circuit is established by  $r_j$ . When  $r_j$  receives the (used) token containing  $M_1$ ,  $r_j$  identifies this token through  $HID = H(PK_{r_j})$ . Anyway, it does not know the sender  $r_i$ . Therefore,  $r_j$  tries to decrypt the fields  $CI, DA, P$  with all its  $k - 1$  proxy keys until it finds the correct key  $k_{ij}$ . Since  $D_{k_{ij}}(DA) \neq S$  (we recall that  $S$  is a fixed string denoting that the token does not contain a message),  $r_j$  has to send the message outside the ring to the destination  $D$  through the Tor system.

Before doing this,  $r_j$  sets the flag  $F = 01$  (reserved) and the field  $HID = H(R)$  where  $R = D_{k_{ij}}(CI)$ . This means that this token is associated with the communication session identified by  $R$ .  $R$  is also stored by  $r_j$  and associated with  $k_{ij}$  in such a way that further tokens can be associated with this communication session. The random  $R$  is also used by  $r_i$  to detect further reserved tokens for this communication session. The other fields are filled with random bits and the token is then forwarded into the ring.

At this point,  $r_j$  can send the message  $M_1 = D_{k_{ij}}(P)$  to the destination  $D$ . To do this, it builds a Tor circuit with destination  $D_{k_{ij}}(DA)$  and sends the message  $M_1$  to  $D$  through this circuit. The construction of the Tor circuit is performed in the standard way, by contacting the Directory Server (DS) and by selecting the entry, middle and exit nodes as illustrated in Section 3.

When  $r_j$  receives a (used) token containing a message  $M_t$  with  $2 \leq t \leq q$ ,  $r_j$  identifies such token through  $HID$  and forwards  $M_t$  to  $D$  through the Tor circuit. The token is set to reserved ( $F = 01$ ) and  $HID$  remains unaltered to the value  $H(R)$ . The other fields are set to random bits, and the token is then forwarded into the ring.

The transmission of the message  $M$  is represented in the sequence diagram in Figure 4.

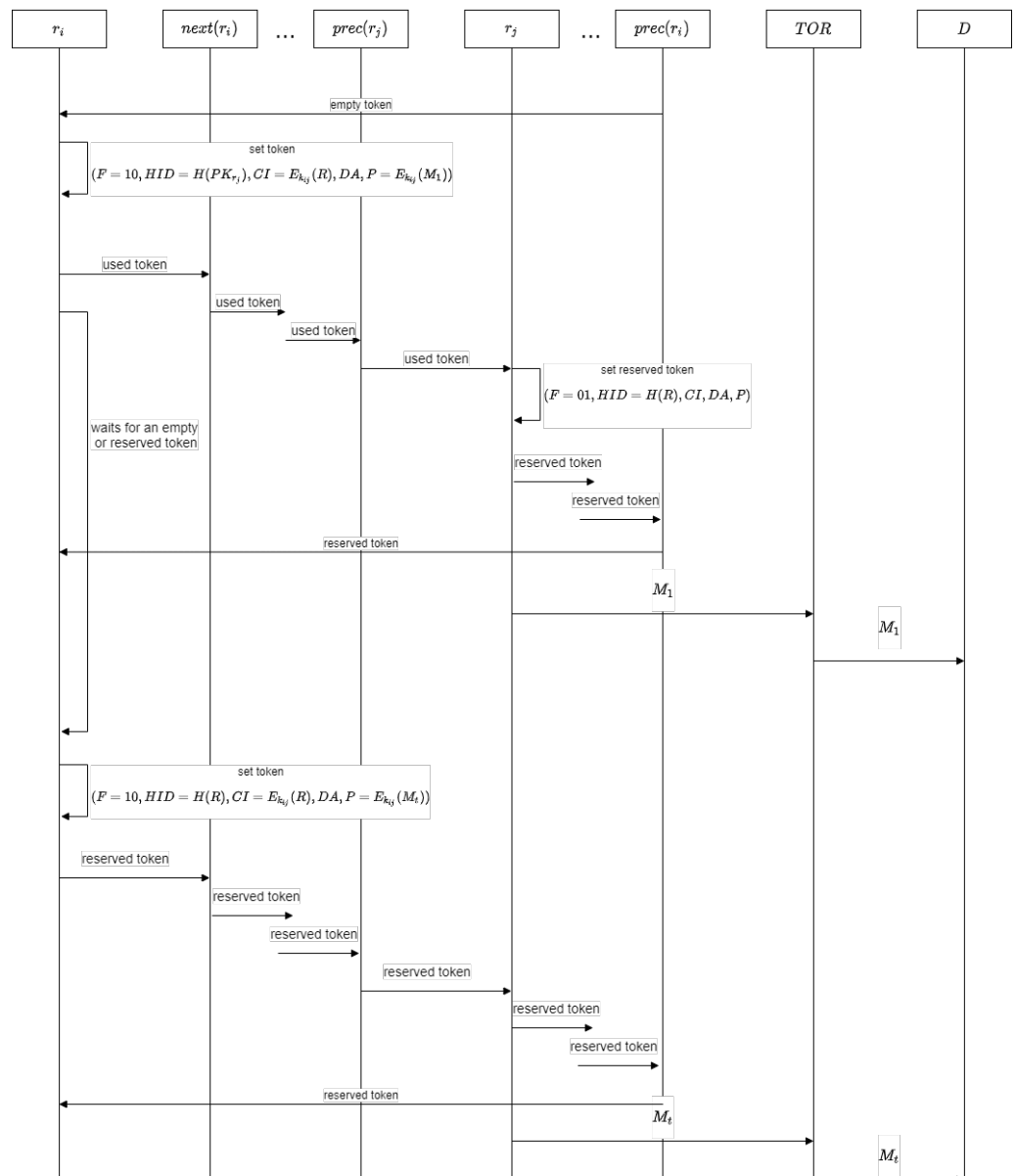


Figure 4. Transmission of the message  $M$ .

#### 4.3.4. Transmission of the Response

When  $r_j$  receives the response  $M'$  (already encrypted by  $D$ ) through the Tor circuit,  $r_j$  injects the response into the ring. Specifically, let  $P'_1, \dots, P'_l$  be the Tor cells including the response  $M'$ , and let denote by  $P_k$  the payload of the cell  $P'_k$  ( $1 \leq k \leq l$ ). For each  $P_k$ ,  $r_j$  waits for either (1) an empty token or (2) a reserved token with  $HID = H(R)$ . The token is

filled as follows.  $F$  is set to 10. Only in the case of an empty token is the field  $HID$  set to  $H(R)$ , and the communication identifier  $CI$ , is derivable by the random  $R$  associated with the current communication session; and then, with this Tor circuit stored by  $r_j$  when the Tor circuit has been established, the field is set properly. That is,  $CI = E_{k_{ij}}(R)$ . The field  $DA$  is filled with random bits. Finally,  $P$  is set to  $E_{k_{ij}}(P_k)$ .

At this point, the token moves through the ring and is identified by  $r_i$  through  $HID$ . When  $r_i$  receives all the tokens containing the block  $P_k$ , it retrieves the entire response  $M'$ . For each of these tokens,  $r_i$  changes the state from used to reserved and forwards the token. Specifically,  $F$  is set to 01,  $HID = H(R)$  is unaltered and the other fields are filled with random bits. These reserved tokens (along with other possible empty tokens) are used by  $r_i$  and  $r_j$  to exchange the other requests/responses associated with the communication session identified by  $R$ .

The transmission of the response  $M'$  is represented in the sequence diagram in Figure 5.

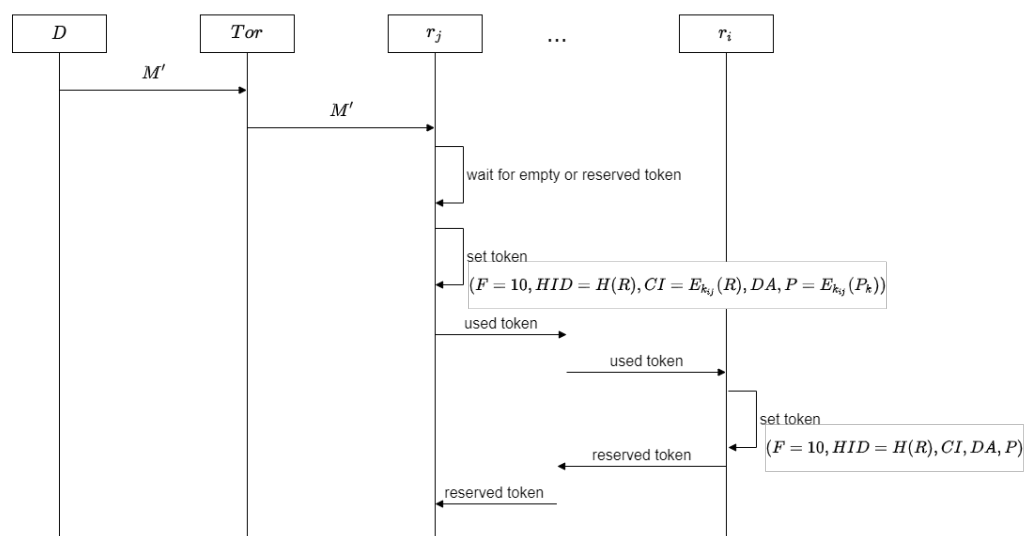


Figure 5. Transmission of the response  $M'$ .

When the communication session ends,  $r_i$  and  $r_j$  perform some actions aimed at emptying the tokens reserved for this session and destroying the Tor circuit. Specifically, for each reserved token with  $HID = H(R)$ ,  $r_i$  fills the token in such a way that  $r_j$  recognizes that they have to be emptied. To do this,  $F$  is set to 10,  $HID = H(R)$  remains unaltered,  $CI$  is set to  $E_{k_{ij}}(R)$ ,  $DA$  is set to the encryption with key  $k_{ij}$  of  $S$  and  $P$  is filled with random bits.

When  $r_j$  receives such token, it retrieves the string  $S$  and recognizes that the session deactivation actions have to be performed. If this token is the first including  $S$ ,  $r_j$  destroys the Tor circuit. For this token and the successive ones, including  $S$ ,  $r_j$  empties them and forwards them into the ring. Specifically,  $F$  is set to 00 and  $HID = H(PK_{next}(r_j))$ . The other fields are filled with random bits.

This process of emptying the tokens and destroying the Tor circuit is represented in the sequence diagram in Figure 6.

To conclude this section, we provide a brief summary, by omitting the technical details of the communication phase. In Figure 7, we sketched a high level graphical representation of this phase.

The sender waits for an empty token, selects a proxy node and fills the token with a message. This token will be injected into the ring, in which it will move until the proxy node is reached. The path of the ring from the sender to the proxy node is represented with a red arrow. Once the proxy node receives the message (possibly, encrypted), it contacts the Directory Server (dashed arrow) to select the entry, middle and exit routers and builds a Tor circuit through them. At this point, the proxy node forwards the message through this Tor circuit until the destination. The latter will provide the response (possibly

encrypted) through the same Tor circuit until the proxy node. Both the ongoing path and the return path are represented by the green arrow in the figure. Finally, when the proxy node receives the response, it waits for a number of empty or reserved tokens and fills them with the response. These tokens are injected into the ring until they reach the originator of the request. The path of the ring from the proxy node to the originator, traversed by the response, is represented with a blue arrow.

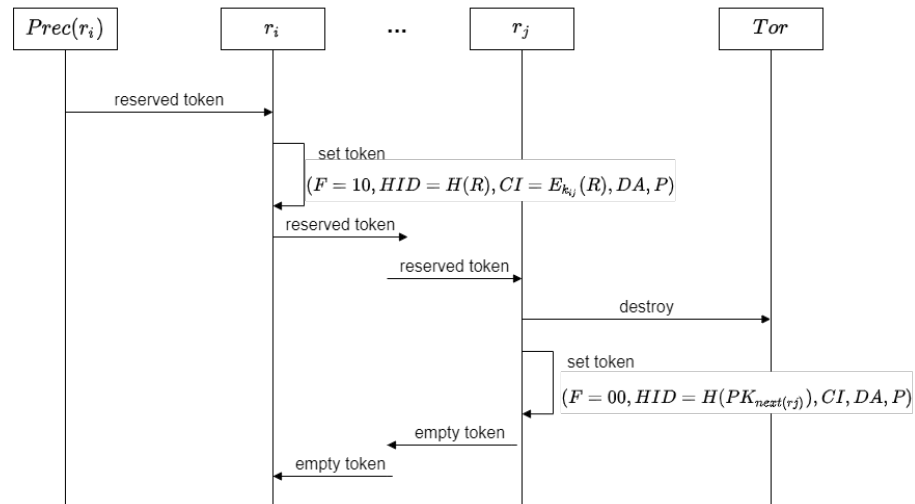


Figure 6. Process of emptying the tokens and destroying the Tor circuit.

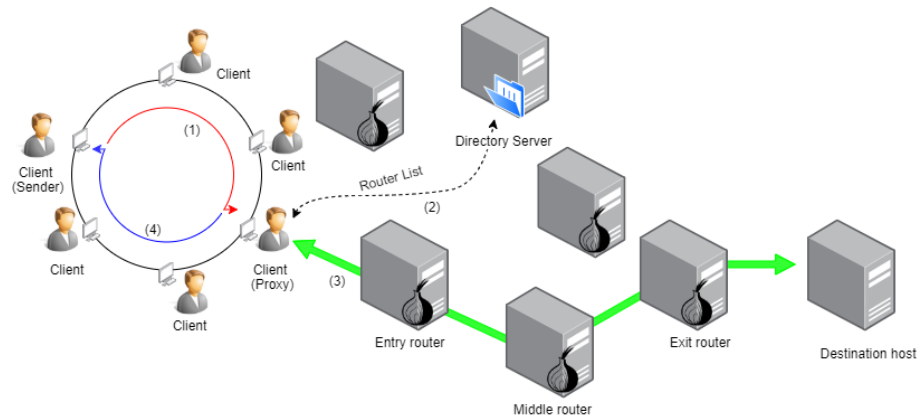


Figure 7. Communication phase.

### 5. Introducing Fault Tolerance into the System

Even though fault tolerance is one of the aspects that is typically missed in anonymous communication networks, we sketch in this section how a certain degree of fault tolerance can be easily introduced in a system based on our protocol. To confirm the above claim, consider the current Tor itself has no fault tolerance at all. Indeed, if a Tor router stops working during a communication, the communication is lost, and there is no a protocol to recover the communication on the fly (indeed, to set a backup Tor circuit is not enough to obtain this goal). As we focus on the part of the proposal that plays the role of *add-on*, with respect to the existing Tor system, we do not consider in this section the Tor communication occurring outside the system, between the proxy node and the destination. Apart from the fact that the fault tolerance of Tor can be considered as an orthogonal problem, it is also true that Tor routers can be considered more stable than standard client nodes involved in the rings.

The basic change we have to introduce to obtain fault tolerance is the notion of a *ring layer*. The ring manager, instead of building simply rings of  $k$  nodes, builds rings of  $k$  layers, each composed of  $j$  nodes. We can figure out that the value of  $j$ , for good fault

tolerance, should be very low (for example, 2 or 3), if we are in a network with a high level of activity. Anyway, higher values of  $j$  do not result in infeasible computation, as we will see next. The nodes of each layer know each other in the sense that they are aware of the reciprocal addresses. With the notation  $r_1, \dots, r_k$ , used earlier for the rings, now we indicate a sequence of layers, such that  $r_i = \{x_1^i, \dots, x_j^i\}$  is a set of  $j$  nodes. Besides the individual public keys of the nodes, there is also a public key per layer, called the *public layer key*. This impacts both the set-up phase and the communication phase. Concerning the set-up phase, some changes occur for the key exchange task. Forward keys are exchanged for each pair  $x_p^i, x_q^{i \% k + 1}$ , ( $1 \leq p, q \leq j$ ). Thus, we have  $j^2$  forward key exchanges per pair of consecutive layers. Instead, by leveraging public layer keys, the pair of keys used as sender key and proxy key  $k_{st}$  and  $k_{ts}$  will be established between layers instead of individual nodes. To do this, the ring manager selects one *representative* node alive per layer and informs each selected node about the other selected nodes (and then about their public keys). Then, the Diffie–Hellman process described in Section 4 happens among these representative nodes. At the end of this process, any representative node has a pair of sender key and proxy key between its layer and any other layer. These keys are exchanged with all the other nodes in the layer. Indeed, in the pre-set-up phase, the nodes of the same layer exchange a symmetric key per pair, by enacting the  $j(j-1)$  Diffie–Hellman processes.

Concerning both the circulation of tokens and the communication task, the only change is that the function *next*, associating to each node of the ring the next node to forward a message, becomes non-deterministic. Specifically, a node in layer  $s$  which has to forward a message, just has to choose one alive node in the layer  $next(s)$  and forwards the message to it. For the proxy node, essentially no change is required, because the encryption is done for the layer, so that any node in the layer is able to decrypt the message and then initiate the Tor circuit. Similar considerations can be made for the response.

To conclude this section, we evaluate our fault-tolerance mechanism from a probabilistic perspective, to allow the correct setting of the parameter  $j$ , once a given reliability probability is fixed. We denote by  $p$  the probability that, at a given instant, a node is alive. We assume  $p$  is the same for each node. Therefore, the probability that, given a layer of  $j$  nodes, at least one node of the layer is alive is  $p' = 1 - (1-p)^j$ . To guarantee reliability (i.e., the communication is not lost), at least one node per level (for the  $k$  levels) has to be alive. Therefore, the probability that the communication succeeds is  $p'' = (1 - (1-p)^j)^k$ . Clearly, it decreases as  $k$  increases and increases as  $j$  increases. Suppose now we set the reliability threshold to a given value  $\tau$ . Then,  $j$  must set in such a way that  $j > \frac{\log(1 - e^{-\frac{\log(\tau)}{k}})}{\log(1-p)}$ .

In Figure 8, we set  $\tau = 0.999$  and show how the ratio  $\frac{j}{k}$  varies for different values of  $p$  and  $k$ .

Observe that the exemplified value chosen for  $\tau$  refers to a very reliable system. Indeed, according to the standard IEC 61508, this value falls into the range of probability of failure on demand (PFD), classifying the system as reliability class SIL 3, which is the second most-reliable class.

As expected, for high values of  $p$ , the number of nodes  $j$  (and then the ratio  $\frac{j}{k}$ ) required to obtain  $\tau = 0.999$  decreases. Regarding  $k$ , as  $k$  increases the absolute value of  $j$  increases but slower than  $k$ . Therefore, the ratio  $\frac{j}{k}$  increases with  $k$ .

To give a practical example, with  $k = 100$  and  $p = 0.9$ , we obtain a ratio  $\frac{j}{k} = 0.05$ , which means that each layer of the ring has to contain only five nodes.



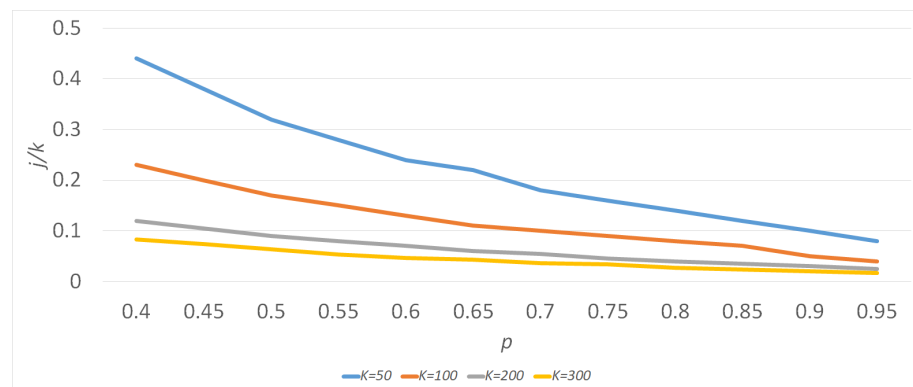


Figure 8. Ratio  $\frac{j}{k}$  as  $k$  and  $p$  vary.

## 6. Computational Complexity

In this section, we discuss the computational complexity of our protocol. We focus on the part of the protocol regarding the ring. Indeed, for the rest of the protocol, involving just a Tor circuit, the reader may refer to the results available in the literature [1].

The communication phase requires, besides the hop-by-hop encryption of the messages (which is standard in any protocol supporting secure communication), the attempted decryptions that the intended proxy node has to perform before sending the message outside the ring. On average, there are  $\frac{k-1}{2}$  decryptions applied only to the first token of a given communication (recall that the size of a token is about 500 bytes). In the worst case, there are  $k - 1$  decryptions. This overhead does not appear relevant, as it regards only the proxy node, and for good privacy levels (e.g.,  $k = 100$ ), the extra time required is small. Observe that the magnitude of an AES encryption/decryption is  $10^2$  Mbytes per second on standard personal computers.

Now, we consider the set-up phase.

First, consider the protocol without fault tolerance (see Section 4). Similarly to the Tor set-up phase, we require  $k$  key exchanges for the forward keys and  $k(k - 1)$  key exchanges for sender/proxy keys. For values of  $k$  guaranteeing a good anonymity level, the cost of this phase is not prohibitive. When fault tolerance is included, we pay a price in terms of complexity of the set-up phase. Indeed, we require  $j(j - 1)$  key exchanges per layer in the pre-set-up phase, and then  $j^2$  forward key exchanges per pair of adjacent layers (executed in parallel) plus  $k(k - 1)$  exchanges for sender/proxy key exchanges. In summary, we increase the previous cost by  $j(j - 1) + j^2$ . Due to the fact that we expect that  $j$  is very small, this computational overhead does not appear as an actual issue for the protocol. Recall that the set-up phase, differently from Tor, is not done for each communication, but it is done to set-up the network, so it can be considered an operation with a long-term lifetime.

## 7. Security Analysis

In this section, we analyze the security of our solution. We start by defining the threat model we consider. We introduce the following assumption:

**Assumption 1 (A1).** Rings are formed in such a way that the background knowledge does not allow the adversary to have more information than sender uniform distribution.

Observe that Assumption A1 is easily satisfied if rings are built among hosts belonging to the same, even large, geographical region.

**Adversary Model (AM).** We consider four types of adversaries.

- **External (E).** In this case, the adversary monitors incoming and outgoing traffic of the DS. In addition, for Ring2Tor, the adversary monitors traffic coming in and going out from the RM.
- **Weak (W).** In this case, the adversary monitors the traffic between a client node and the entry Tor router. In Tor, the client node corresponds to the OP. To be fair, for

Ring2Tor, we allow the weak adversary to monitor all the traffic between the client nodes and the traffic between the client node playing as a proxy and the entry Tor router.

- **Strong (S).** In this case, the adversary monitors the traffic between a client node and the entry Tor router and the traffic between the exit Tor router and the destination host. For Ring2Tor, in addition, the adversary can monitor all the traffic between the client nodes.
- **Global (G).** In this case, the adversary monitors all the traffic of the network.

Furthermore, for all the four adversaries, regarding Ring2Tor, we enable another capability: the adversary knows the entire composition of the rings.

Observe that the capabilities of Global, Strong and Weak adversaries are in order (i.e., Global is stronger than Strong and Strong is stronger than Weak). Furthermore, Global is stronger than External.

Both the External adversary and the Weak adversary model refer to a very feasible case in which an entity is able to control just an autonomous system. The feasibility of the External adversary can be contrasted by distributing the DS and the RM. The Strong adversary is a weak form of the Global adversary, because the autonomous systems of entry router and exit router can be very far from each other and even be in different continents [14]. The Global adversary is the standard global passive adversary.

**Security properties.** We analyze two security properties (see Section 3): (1) Sender anonymity (SA); (2) Relationship anonymity (RA).

In the following analysis, we discuss how Tor and Ring2Tor behave with respect to the security properties in the four adversary models. The results of the analysis are summarized in Table 1. First, we give a preliminary basic result in the following lemma.

**Lemma 1.** *In Ring2Tor, a ring of size  $k$  is a sender with an anonymity set of size  $k$  against the Global adversary.*

**Proof of Lemma 1.** Due to the hop-by-hop probabilistic encryption mechanism that is used to move tokens inside the ring, the only point of the ring from which the adversary can draw some information more than a random guess to identify a sender is the proxy node. Indeed, this is the only point of the ring in which the possible state transitions of a token could be in principle related to the observable incoming or outgoing traffic in/out of the proxy. Transitions occurring in other points are not identifiable with probability higher than  $\frac{1}{k}$ . Since reserved and used tokens cannot be filled by other client nodes different from the sender (associated with the reserved tokens), the only possibility for the adversary to identify a sender anonymity set of size less than  $k$  is to detect an empty token outgoing from a node and track it until it reaches a proxy node, which sends a message outside the ring before doing less than  $k$  steps. The only event in which the adversary can guess that a token is emptied is when a proxy node, say  $r_x$ , dismisses a Tor circuit. Indeed, according to the protocol, there is no other case in which tokens are emptied. However,  $r_x$  sets the field  $HID$  to  $H(PK_{next(r_x)})$ , and this means that such a token moves around the entire ring (in which it is, possibly, filled) before reaching  $next(r_x)$ , which possibly builds a Tor circuit outside the ring. Therefore, we can argue that the sender anonymity set is not always larger than  $k$ , even for the Global passive adversary. The proof is then concluded.  $\square$

The above lemma is the basis for the fulfillment of the security properties stated above for Ring2Tor.

This is proven through the following theorems. The first theorem states that Tor does not guarantee SA against any adversary. This corresponds to the first four fields of the first row of Table 1.

**Theorem 1.** *In Tor, any adversary breaks SA with probability 1.*

**Proof of Theorem 1.** Consider the External adversary. Since it observes the traffic intended to the DS, it receives the request of the sender and then the sender is identified. Since the Global adversary has the same capabilities as the External adversary, SA does not hold against it. Now, we consider the Weak adversary able to observe the traffic between the sender and the entry Tor router. Clearly, W identifies the sender. The Strong adversary has the same capabilities as the Weak adversary. The proof is then concluded. □

Now, we prove that Ring2Tor guarantees that a sender can be identified (by any adversary) with probability  $\frac{1}{k}$ . This corresponds to the first four fields of the second row of Table 1.

**Theorem 2.** In Ring2Tor, any adversary breaks SA with probability  $\frac{1}{k}$ .

**Proof of Theorem 2.** Consider the Global adversary G. By Lemma 1, it can identify the sender with a probability not higher than  $\frac{1}{k}$ . Since G is stronger than all the other adversaries (i.e., S, W, and E), we conclude that for those three adversaries also, SA is broken with a probability not higher than  $\frac{1}{k}$ . □

**Table 1.** Comparison between Tor (T) and Ring2Tor(R2T). Shown are the probabilities of the adversaries breaking the properties SA and RA.

	SA				RA			
AM	E	W	S	G	E	W	S	G
T	1	1	1	1	$\frac{1}{n_d}$	$\frac{1}{n_d}$	1	1
R2T	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{k}$	$\frac{1}{n_d \cdot k}$	$\frac{1}{n_d \cdot k}$	$\frac{1}{k}$	$\frac{1}{k}$

Now, we have to consider the remaining fields of Table 1 regarding relationship anonymity. These are covered by the following two theorems.

**Theorem 3.** Let  $n_d$  be the size of the recipient anonymity set. In Tor, the External and Weak adversary break RA with probability  $\frac{1}{n_d}$ . Furthermore, the Strong and Global adversary break RA with probability 1.

**Proof of Theorem 3.** Consider the External adversary. By Theorem 1, it identifies the sender SN of a communication with probability 1. Anyway, E has no information about the recipient R of such a communication. Therefore, E (without further knowledge) identifies that SN communicates with R only with the smallest probability, i.e.,  $\frac{1}{n_d}$ .

Similarly, the Weak adversary identifies the sender with probability 1, but has no information about the recipient. Therefore, RA is broken with probability  $\frac{1}{n_d}$ .

Consider the Strong adversary S. Since it monitors the outgoing traffic from the exit Tor router, it can identify the recipient R of a communication with probability 1. Since S also monitors the traffic between the sender SN and the entry Tor router, it can perform traffic analysis attacks [14] and identifies that SN communicates with R with probability 1. The Global adversary has the same power as the Strong adversary. The proof is then concluded. □

**Theorem 4.** Let  $n_d$  be the size of the recipient anonymity set. In Ring2Tor, the External and Weak adversary break RA with probability  $\frac{1}{n_d \cdot k}$ . Furthermore, the Strong and Global adversary break RA with probability  $\frac{1}{k}$ .

**Proof of Theorem 4.** Since SA implies RA [7], by Theorem 2, it follows that RA can be broken with a probability not higher than  $\frac{1}{k}$  by any adversary. Consider now the adversaries E and W. Even though they can identify the sender with a probability not higher than  $\frac{1}{k}$ ,

they do not have any information about the recipient. Therefore, they can only guess the recipient among all the possible recipients of the network  $n_d$ . Therefore, for  $E$  and  $W$ ,  $RA$  is broken with a probability not higher than  $\frac{1}{n_d \cdot k}$ . For the other adversaries (i.e.,  $S$  and  $G$ ), the above upper bound of the success probability cannot be decreased, because both  $S$  and  $G$  are able to identify the recipient, so that the probability of breaking  $RA$  is the same as the probability of breaking  $SA$ . The proof is then concluded.  $\square$

This ends the security analysis. As is evident by Table 1, the benefit in terms of security of Ring2Tor can be measured as a multiplicative factor  $k$ , increasing the degree of anonymity provided by Tor both for  $SA$  and  $RA$ .

## 8. Experiments

In this section, we provide experimental validation of Ring2Tor. Specifically, our aim is to show that the network performance is not compromised by the adoption of our protocol. Our analysis was performed through the network simulator NS3 [39]. We simulated an overlay network with rings of size  $k = 50$  and with no fault tolerance (clearly, fault tolerance has no impact on the communication-phase performance, only on the set-up phase). Tor routers were set as separated network nodes. Regarding the links between the Tor routers, we set a delay such that the total time to perform a download of 50KB was about 1.5 s, which represents the actual time (as of October 2021) taken to download a file of this size in the real-life Tor network [40]. The resulting delay is then 150 ms.

The above considerable delays reflect the fact that, according to the current Tor-router-selection algorithm, no two routers in the same circuit belong to the same class B network (/16 subnet) or the same family [41]. Regarding the link between client nodes, we set a delay of 10 ms, capturing that the purpose of the node-selection algorithm—to form rings is the opposite of that of Tor router selection—to obtain homogeneity among nodes in a ring (and thus an effective anonymity set). Therefore, nodes belonging to the same ring are geographically close to each other.

We used an http traffic generation model that simulates web browsing traffic according to the specification suggested by 3GPP2 [42]. We focus our analysis on three metrics: (communication) latency, (traffic) overhead and throughput. The communication latency is defined by the application layer, as it measures the time between the instant at which the sender sends an http request and the instant at which it receives the complete response. Observe that, to be fair, we did not consider as initial time the instant in which the sender received an available token, but the instant in which the http client generated the request. As traffic overhead, we took the average ratio between the number of empty tokens circulating in the ring and the total number of circulating tokens. Finally, the throughput was defined as per usual—that is, the average exploited data rate per node. The results are reported in Figures 9–13.

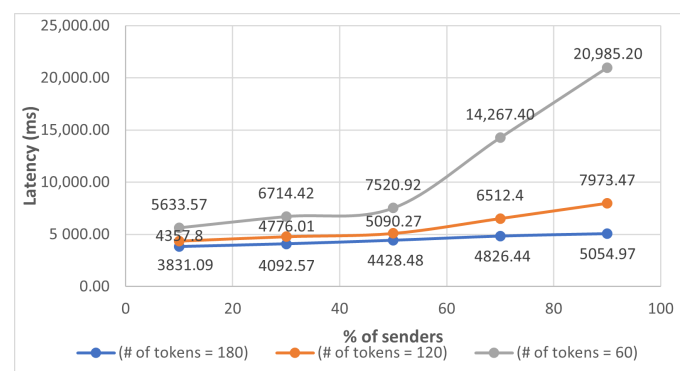


Figure 9. Latency vs. percentage of senders in Ring2Tor.

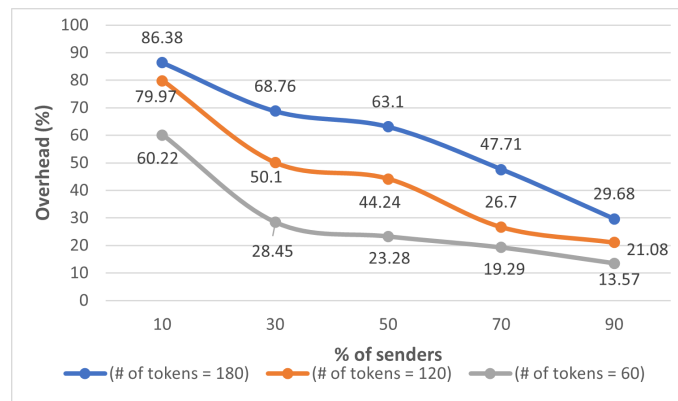


Figure 10. Overhead vs. percentage of senders in Ring2Tor.

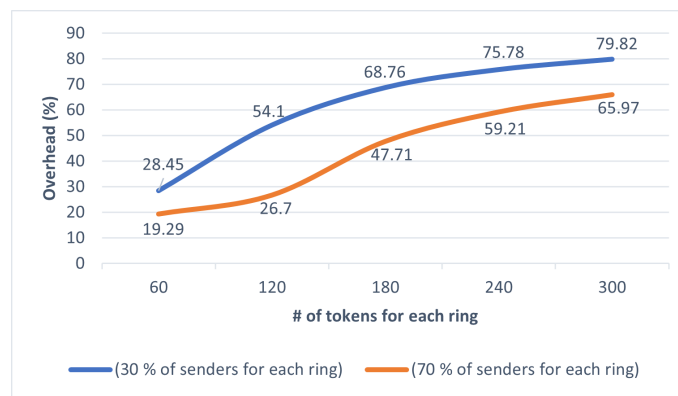


Figure 11. Overhead vs. Number of tokens in Ring2Tor.

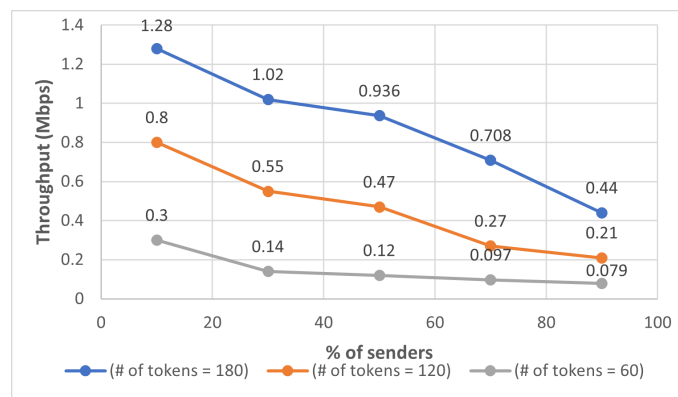


Figure 12. Throughput vs. percentage of senders in Ring2Tor.

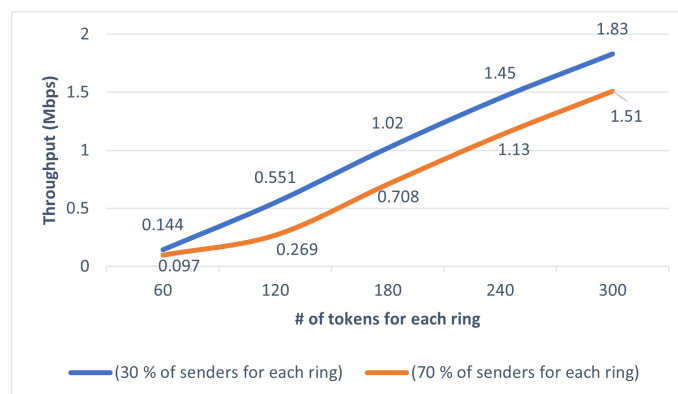


Figure 13. Throughput vs. numbers of tokens in Ring2Tor.



The plot in Figure 9 shows that our solution introduces an acceptable latency for different sender percentages. Specifically, when the percentage of senders is less than 50, the latency ranges from 3.8 s to nearly 7 s, which are values not too far from those found for Tor [43], considering the results obtained for low-volume http traffic.

For high volumes, the difference between Ring2Tor and Tor increases, because Tor's performance improves. However, the absolute values of latency experimented for Ring2Tor for realistic http traffic can be considered acceptable. As expected, the latency increases as the sender percentage does, since, in the rings, many filled (reserved and used) tokens move. On the other hand, the latency decreases as the number of tokens moving in the ring increases.

The plots in Figures 10 and 11 can be used to set the number of tokens that a generator needs to maintain in a ring. Indeed, by fixing the maximum percentage of overhead tolerable and the percentage of senders in the rings, we can find the minimum number of tokens. For example, if the maximum overhead is set to 27%, even with a high percentage of senders equal to 70%, we obtain a latency of 6.5 seconds by setting 120 tokens in the ring.

As expected, the overhead decreases as the percentage of senders increases, because there will be more filled tokens circulating into the ring. Moreover, as the total number of tokens circulating into the ring increases, with the same percentage of senders, the overhead increases since a greater number of empty tokens will circulate into the ring.

Finally, Figures 12 and 13 show that also the values of throughput are acceptable. As expected, regardless the number of tokens, the throughput decreases as the percentage of senders increases. This happens since when the percentage of senders increases, the number of empty (available) tokens decreases and then, each sender has to wait for a longer time before sending a message. This reduces the throughput of the senders. On the other hand, as the number of tokens increases, also the number of empty (available) increases and then, with the same percentage of senders, the throughput experimented by a single sender increases.

Even though the amount of traffic overhead could appear high, we have to consider that we are dealing with an inherently difficult task, which is the resistance to a global passive adversary. It is widely recognized in the literature that a high traffic overhead is the price we have to pay in any anonymous routing protocol to achieve the above goal [28]. As a matter of fact, our protocol has a significant advantage with respect to the standard way of hiding communication against a global passive adversary. The standard way is indeed to use mixnets [29] with bi-directional cover traffic in any link of the overlay network. Instead, in our approach, cover traffic is only 1-directional and the circular overlay network, differently from mixnets, does not produce overhead amplification. To better understand this point, consider a simplified yet general model of mixnets taken from [32]. Here, as anticipated earlier, we need to enable bi-directional cover traffic over any link of the overlay network in such a way that the fan-out mechanism increases the cardinality of the anonymity set exponentially with the length of the communication path. Indeed, if we have even a simple mixnet with a degree of mixing of 2 (i.e., the traffic of two senders is mixed into one receiver at each step), for a communication path of length  $l$ , the anonymity set has cardinality  $2^l$ . For minimum degree of nodes (which is 3, to enable the fan-out mechanism) and  $k$  nodes, the cover traffic is  $2 \cdot 3 \cdot k$  (recall that the traffic must be bi-directional). Instead, in our protocol, the cover traffic involving  $k$  nodes in a ring is just  $k$ , according to the topology with no branch of the route and the fact that the traffic is 1-directional. Therefore, we reduce cover traffic by a multiplicative factor equal to 6. Observe that, in these approaches, the cover traffic is the total traffic of the network. However, when the cover traffic does not embed real traffic, it represents an overhead.

To better support the above analysis, we performed a number of experiments by implementing this simple model of P2P mixnet with 50 nodes, each with degree 3. In particular, we replicated the same simulation conditions in NS3 as those used for Ring2Tor (same traffic pattern, same link delay and same number of nodes). Furthermore, we

measured the rate of the circulating tokens in Ring2Tor and used this value to set the rate of the total traffic for a single link of the mixnet.

In Figure 14, we show that the ratio between the cover traffic of the mixnet and the cover traffic of Ring2Tor is very close to 6 (the slight difference comes from possible imprecision of the rate setting).

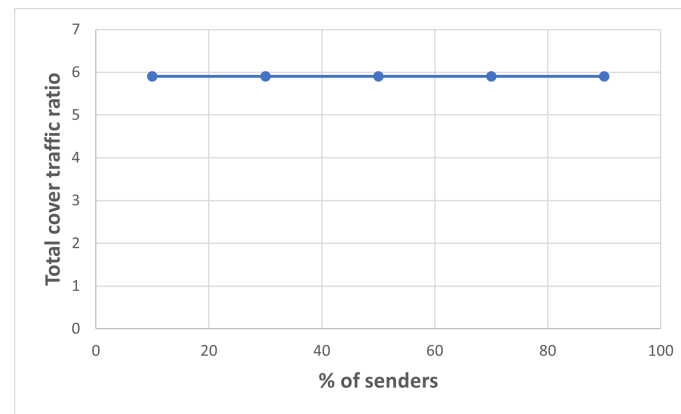


Figure 14. Ratio between the cover traffic of mixnet and Ring2Tor.

The benefits of our solution in terms of traffic overhead compared with the mixnet are highlighted in Figure 15. Therein, we consider Ring2Tor with 180 tokens circulating in each ring. For the mixnet, we obtain a very high value of traffic overhead (approximately equal to 99%). Observe that it decreases very slightly with the percentage of senders. This can be explained by considering that, for the bursting http traffic, the high volume of total cover traffic is always dominant, even when many nodes are senders.

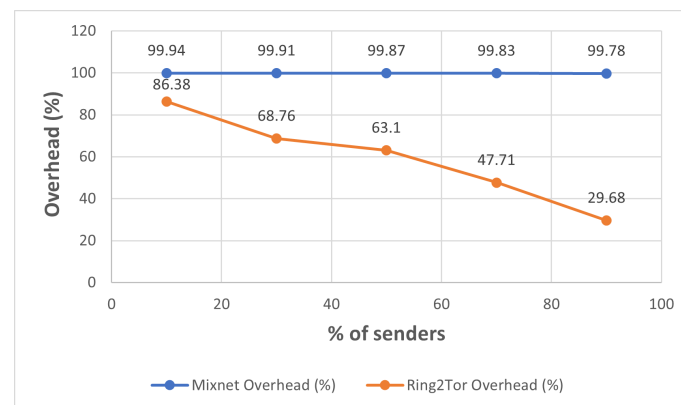


Figure 15. Overhead vs. percentage of senders in mixnet and Ring2Tor.

We expect that the benefits in terms of overhead have a price in terms of latency, as the communication path is in general much shorter in mixnets than in Ring2Tor. Indeed, in Ring2Tor, the request and the associated response go through  $O(k)$  hops, but in mixnet, through  $O(\log k)$  nodes. It is important to understand whether, for a significant privacy level (i.e., the cardinality of the anonymity set), the above price is tolerable or not. To do this, we performed an experiment on latency. The results are reported in Figure 16. From them, we can conclude that the latency of Ring2Tor is higher (as expected), but within a range of tolerability, for the considered application setting.

In summary, we can conclude that our protocol represents a good trade-off between latency and traffic overhead, when resistance to a global passive adversary should be achieved.

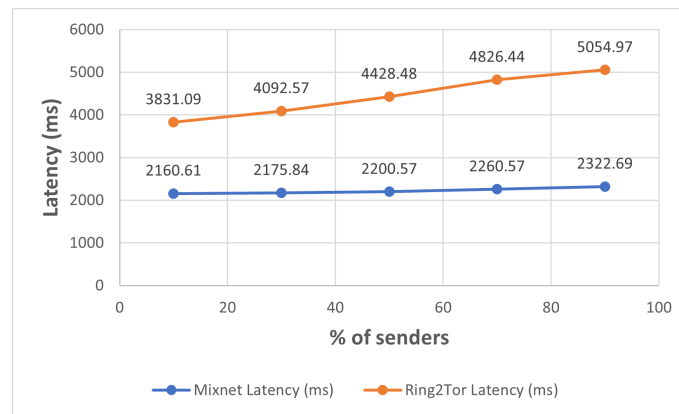


Figure 16. Latency vs. percentage of senders in mixnet and Ring2Tor.

## 9. Conclusions

In this paper, we proposed the protocol Ring2Tor achieving sender anonymity in Tor, and then relationship anonymity, against a global passive adversary. We conducted a security analysis, showing that the ring-based approach guarantees a sender an anonymity set the size of the ring, thereby allowing us to achieve our goal. The protocol includes also a certain degree of fault tolerance to consider the case in which not all nodes are alive and collaborative. Moreover, a computational complexity analysis of the solution was also provided. As typically happens for every  $k$ -anonymity-based approach to achieve privacy, a crucial point is to establish which is the right value of  $k$  to effectively cover the adversary monitoring. Evidently, the higher the value of  $k$ , the stronger the protection. However, an interesting question to pose is that there is a threshold to reach. The answer to this question can be acquired only from a risk-based point of view. Indeed, to guarantee  $k$ -anonymity, as our approach does, we have to provide the risk analyst with a concrete way to estimate the probability for the adversary of re-identification of a possible target. This is necessary to process risk analysis, together with the evaluation of the impact of a similar event. Therefore, the threshold may directly derive from the requirements in terms of risk we can set, depending on the application setting. In a key paper [8], some considerations about this aspect are given. Therein, the authors say that, “ $k$ -anonymity is still sufficient for a variety of applications. For example, in the United States legal system, 2-anonymity would be enough to cast *reasonable doubt*, thus invalidating a criminal charge, while 3-anonymity would be enough to invalidate a civil charge, in the absence of other evidence”.

Once the specific application scenario is fixed, the risk threshold can thus established. Then, from the evaluation of the impact of a re-identification incident, the maximum allowed probability of the incident can be derived (also on the basis of the expected capabilities of the adversary). Therefore, the right privacy level  $k$  can be set.

Obviously, the chosen value of  $k$  has direct impact on the network’s performance. The experimental validation highlighted that it is possible to configure the network in order to obtain acceptable values of overhead, latency or throughput, depending on the requirements. We can argue that the price in terms of network performance to obtain our strong anonymity goal is tolerable, when privacy needs are high priority. We traced the route for further investigation in the direction of more sophisticated setting of the network configuration (e.g., number of tokens, size of the rings), by enabling suitable adaptivity to better control network performance in the dynamic case.

Another direction to investigate as future work is represented by a formal validation analysis of our solution regarding dependability and security requirements. For example, reference [44] proposes an approach to validating solutions involving dynamic changes. It appears very suitable for our protocol, since it requires several message exchanges and sequential steps (generation, filling, and emptying of tokens). Another interesting framework that could be applied during the security design and development of our protocol is [45].

Furthermore, in the context of security, the analysis proposed in Section 7, combined with approach proposed in [46,47], can represent a starting point to derive specific security patterns applicable to anonymous communication networks, and thus, to our case. These patterns may involve: (1) the requirement phase, both in terms of analysis process patterns and model based patterns; (2) the design phase, by considering the design of security properties, the bridge between security design patterns and security properties and the proper domain-specific design patterns; (3) the implementation phase of the software we have to install in the nodes and in the ring manager, through secure programming guidelines, attack pattern catalog definition and secure refactoring.

**Author Contributions:** Conceptualization, F.B., V.D.A., M.F.I. and C.L.; methodology, F.B., V.D.A., M.F.I. and C.L.; software, S.L.; validation, F.B., V.D.A., M.F.I., C.L. and S.L.; formal analysis, F.B. and V.D.A.; investigation, F.B., V.D.A., M.F.I. and C.L.; resources, S.L.; data curation, S.L.; writing—original draft preparation, F.B., V.D.A., M.F.I. and C.L.; writing—review and editing, F.B., V.D.A., M.F.I., C.L. and S.L.; visualization, S.L.; supervision, F.B.; project administration, F.B. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** This paper is partially supported by Project POR FESR/FSE 14/20 Line A (Action 10.5.6) and Line B (Action 10.5.12).

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

AM	Adversary Model
CI	Communication Identifier
DA	Destination Address
DH	Diffie–Hellman
DS	Directory Server
E	External Adversary
F	Flag
G	Global Adversary
H	Hash Value
HID	Hashed Identifier
OP	Onion Proxy
OR	Onion Router
P	Payload
P2P	Peer To Peer
PDH	Public Diffie–Hellman
PFD	Probability of Failure
R	Random
RA	Relationship Anonymity
RM	Ring Manager
S	Strong Adversary
SA	Sender Anonymity
W	Weak Adversary
SIL	Safety Integrity Level
T	Tor
R2T	Ring2Tor

## References

1. Syverson, P.; Dingedine, R.; Mathewson, N. Tor: The Second-Generation Onion Router. In Proceedings of the Usenix Security, San Diego, CA, USA, 9–13 August 2004; pp. 303–320.
2. Goldschlag, D.M.; Reed, M.G.; Syverson, P.F. Hiding Routing Information. In Proceedings of the International Workshop on Information Hiding, Cambridge, UK, 30 May–1 June 1996; Springer: Berlin/Heidelberg, Germany, 1996; pp. 137–150.
3. Reiter, M.K.; Rubin, A.D. Crowds: Anonymity for Web Transactions. *ACM Trans. Inf. Syst. Secur. (TISSEC)* **1998**, *1*, 66–92. [[CrossRef](#)]
4. O’Gorman, G.; Blott, S. Large Scale Simulation of Tor. In Proceedings of the Annual Asian Computing Science Conference, Doha, Qatar, 9–11 December 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 48–54.
5. Karunanayake, I.; Ahmed, N.; Malaney, R.; Islam, R.; Jha, S. Anonymity with Tor: A Survey on Tor Attacks. *arXiv* **2020**, arXiv:2009.13018.
6. Palmieri, F. A Distributed Flow Correlation Attack to Anonymizing Overlay Networks Based on Wavelet Multi-Resolution Analysis. *IEEE Trans. Depend. Secur. Comput.* **2019**, *18*, 2271–2284. [[CrossRef](#)]
7. Pfitzmann, A.; Hansen, M. A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management. 2010. (Version 0.33 April 2010), Technical Report, TU Dresden and ULD Kiel. Available online: [http://dud.inf.tu-dresden.de/Anon\\_Terminology.shtml](http://dud.inf.tu-dresden.de/Anon_Terminology.shtml) (accessed on 21 May 2021).
8. Von Ahn, L.; Bortz, A.; Hopper, N.J. k-Anonymous Message Transmission. In Proceedings of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA, 27–30 October 2003; pp. 122–130.
9. Hirt, A.; Jacobson, M.; Williamson, C. Taxis: Scalable Strong Anonymous Communication. In *Proceedings of the 2008 IEEE International Symposium on Modeling, Analysis and Simulation of Computers and Telecommunication Systems, Baltimore, MD, USA, 8–10 September 2008*; IEEE Computer Society: Washington, DC, USA, 2008; pp. 1–10.
10. Beimel, A.; Dolev, S. Buses for Anonymous Message Delivery. *J. Cryptol.* **2003**, *16*, 25–39
11. Burnside, M.; Keromytis, A.D. Low Latency Anonymity with Mix Rings. In Proceedings of the International Conference on Information Security, Samos Island, Greece, 30 August–2 September 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 32–45.
12. Salo, J. *Recent Attacks on Tor*; Aalto University: Espoo, Finland 2010.
13. Erdin, E.; Zachor, C.; Gunes, M.H. How to Find Hidden Users: A Survey of Attacks on Anonymity Networks. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2296–2316. doi:10.1109/COMST.2015.2453434. [[CrossRef](#)]
14. Basyoni, L.; Fetais, N.; Erbad, A.; Mohamed, A.; Guizani, M. Traffic Analysis Attacks on Tor: A Survey. In *Proceedings of the 2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT), Doha, Qatar, 2–5 February 2020*; IEEE Computer Society: Washington, DC, USA, 2020; pp. 183–188.
15. Edman, M.; Yener, B. On Anonymity in an Electronic Society: A Survey of Anonymous Communication Systems. *ACM Comput. Surv. (CSUR)* **2009**, *42*, 1–35. [[CrossRef](#)]
16. Murdoch, S.J.; Danezis, G. Low-Cost Traffic Analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy (S&P’05), Oakland, CA, USA, 8–11 May 2005*; IEEE Computer Society: Washington, DC, USA, 2005; pp. 183–195.
17. Levine, B.N.; Reiter, M.K.; Wang, C.; Wright, M. Timing Attacks in Low-Latency Mix Systems. In *Proceedings of the International Conference on Financial Cryptography, Key West, FL, USA, 9–12 February 2004*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 251–265.
18. Syverson, P.; Tsudik, G.; Reed, M.; Landwehr, C. Towards an Analysis of Onion Routing Security. In *Designing Privacy Enhancing Technologies*; Springer: Berlin/Heidelberg, Germany, 2001; pp. 96–114.
19. Gilad, Y.; Herzberg, A. Spying in the Dark: TCP and Tor Traffic Analysis. In *Proceedings of the International Symposium on Privacy Enhancing Technologies Symposium, Vigo, Spain, 11–13 July 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 100–119.
20. Rochet, F.; Pereira, O. Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols. *Proc. Priv. Enhanc. Technol.* **2018**, *2018*, 27–46. [[CrossRef](#)]
21. Iacovazzi, A.; Elovici, Y. Network Flow Watermarking: A Survey. *IEEE Commun. Surv. Tutor.* **2016**, *19*, 512–530. [[CrossRef](#)]
22. Snader, R.; Borisov, N. A Tune-up for Tor: Improving Security and Performance in the Tor Network. In Proceedings of the NDSS, San Diego, CA, USA, 10–13 February 2008; Volume 8, p. 127.
23. Johnson, A.; Jansen, R.; Hopper, N.; Segal, A.; Syverson, P. PeerFlow: Secure Load Balancing in Tor. *PoPETs* **2017**, *2017*, 74–94. [[CrossRef](#)]
24. Bauer, K.S.; Sherr, M.; Grunwald, D. ExperiTor: A Testbed for Safe and Realistic Tor Experimentation. In Proceedings of the CSET, San Francisco, CA, USA, 8 August 2011.
25. Panchenko, A.; Pimenidis, L.; Renner, J. Performance Analysis of Anonymous Communication Channels Provided by Tor. In *Proceedings of the 2008 Third International Conference on Availability, Reliability and Security, Barcelona, Spain, 4–7 March 2008*; IEEE Computer Society: Washington, DC, USA, 2008; pp. 221–228.
26. Komlo, C.H.; Mathewson, N.; Goldberg, I. Walking Onions: Scaling Anonymity Networks while Protecting Users. In Proceedings of the 29th USENIX Security Symposium (USENIX Security 20), Boston, MA, USA, 12–14 August 2020; pp. 1003–1020.
27. Cangialosi, F.; Levin, D.; Spring, N. Ting: Measuring and Exploiting Latencies Between All Tor Nodes. In Proceedings of the 2015 Internet Measurement Conference, Tokyo, Japan, 28–30 October 2015; pp. 289–302.
28. Danezis, G.; Diaz, C. *A Survey of Anonymous Communication Channels*; Technical Report, Technical Report MSR-TR-2008-35; Microsoft Research: Cambridge, UK 2008.



29. Chaum, D.L. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Commun. ACM* **1981**, *24*, 84–90. [[CrossRef](#)]
30. Young, A.L.; Yung, M. The drunk motorcyclist protocol for anonymous communication. In *Proceedings of the 2014 IEEE Conference on Communications and Network Security, San Francisco, CA, USA, 29–31 October 2014*; IEEE Computer Society: Washington, DC, USA, 2014; pp. 157–165.
31. Wang, W.; Motani, M.; Srinivasan, V. Dependent link padding algorithms for low latency anonymity systems. In *Proceedings of the 15th ACM Conference on Computer and Communications Security, Tokyo, Japan, 18–20 March 2008*; pp. 323–332.
32. Freedman, M.J.; Morris, R. Tarzan: A Peer-to-Peer Anonymizing Network Layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18–22 November 2002*; pp. 193–206.
33. Le Blond, S.; Choffnes, D.; Zhou, W.; Druschel, P.; Ballani, H.; Francis, P. Towards Efficient Traffic-Analysis Resistant Anonymity Networks. *ACM SIGCOMM Comput. Commun. Rev.* **2013**, *43*, 303–314. [[CrossRef](#)]
34. Kotzanikolaou, P.; Chatzisoifroniou, G.; Burmester, M. Broadcast anonymous routing (BAR): Scalable real-time anonymous communication. *Int. J. Inf. Secur.* **2017**, *16*, 313–326. [[CrossRef](#)]
35. Chen, C.; Asoni, D.E.; Perrig, A.; Barrera, D.; Danezis, G.; Troncoso, C. TARANET: Traffic-Analysis Resistant Anonymity at the Network Layer. In *Proceedings of the 2018 IEEE European Symposium on Security and Privacy (EuroS&P), London, UK, 24–26 April 2018*; IEEE Computer Society: Washington, DC, USA, 2018; pp. 137–152.
36. Buccafurri, F.; De Angelis, V.; Idone, M.F.; Labrini, C. Anonymous Short Communications over Social Networks. In *Proceedings of the EAI SecureComm 2021—17th EAI International Conference on Security and Privacy in Communication Networks, Virtual Event, 6–9 September 2021*; Springer: Berlin/Heidelberg, Germany 2021.
37. Shirazi, F.; Simeonovski, M.; Asghar, M.R.; Backes, M.; Diaz, C. A Survey on Routing in Anonymous Communication Protocols. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–39. [[CrossRef](#)]
38. Buccafurri, F.; De Angelis, V.; Idone, M.F.; Labrini, C. Wip: An Onion-Based Routing Protocol Strengthening Anonymity. In *Proceedings of the 2021 IEEE 22nd International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), Pisa, Italy, 7–11 June 2021*; pp. 231–235.
39. ns-3—Network Simulator 3. 2021. Available online: <https://www.nsnam.org/documentation/> (accessed on 21 May 2021).
40. TorPerformance. 2021. Available online: <https://metrics.torproject.org/torperf.html> (accessed on 21 May 2021).
41. AlSabah, M.; Goldberg, I. Performance and Security Improvements for Tor: A Survey. *ACM Comput. Surv. (CSUR)* **2016**, *49*, 1–36. [[CrossRef](#)]
42. 3GPP2-TSGC5. *HTTP, FTP and TCP Models for 1xEV-DV Simulations*; 3GPP2: Arlington, VA, USA, 2001.
43. Keita, B. Experimental Evaluation of the Impact of Tor Latency on Web Browsing. 2021. Available online: <https://witestlab.poly.edu/blog/latency-tor/> (accessed on 21 May 2021).
44. Muñoz, A.; Maña, A.; Serrano, D. AVISPA in the Validation of Ambient Intelligence Scenarios. In *Proceedings of the 2009 International Conference on Availability, Reliability and Security, Fukuoka, Japan, 16–19 March 2009*; IEEE Computer Society: Washington, DC, USA, 2009; pp. 420–426.
45. Serrano, D.; Ruíz, J.F.; Muñoz, A.; Maña, A.; Armenteros, A.; Crespo, B.G.N. Development of Applications Based on Security Patterns. In *Proceedings of the 2009 Second International Conference on Dependability, Athens, Greece, 18–23 June 2009*; IEEE Computer Society: Washington, DC, USA, 2009; pp. 111–116.
46. Yoshioka, N.; Washizaki, H.; Maruyama, K. A survey on security patterns. *Prog. Inf.* **2008**, *5*, 35–47. [[CrossRef](#)]
47. Schumacher, M.; Fernandez-Buglioni, E.; Hybertson, D.; Buschmann, F.; Sommerlad, P. *Security Patterns: Integrating Security and Systems Engineering*; John Wiley & Sons: Hoboken, NJ, USA, 2013.