DUO-ONIONS AND HYDRA-ONIONS – FAILURE AND ADVERSARY RESISTANT ONION PROTOCOLS *

Jan Iwanik, Marek Klonowski, and Mirosław Kutyłowski
iwanik@im.pwr.wroc.pl, klonowsk@im.pwr.wroc.pl, Miroslaw.Kutylowski@pwr.wroc.pl
Institute of Mathematics, Wrocław Univ. of Technology, ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland

Abstract A serious weakness of the onion protocol, one of the major tools for anonymous communication, is its vulnerability to network failures and/or an adversary trying to break the communication. This is facilitated by the fact that each message is sent through a path of a certain length and a failure in a single point of this path prohibits message delivery. Since the path cannot be too short in order to offer anonymity protection (at least logarithmic in the number of nodes), the failure probability might be quite substantial.

The simplest solution to this problem would be to send many onions with the same message. We show that this approach can be optimized with respect to communication overhead and resilience to failures and/or adversary attacks. We propose two protocols: the first one mimics K independent onions with a single onion. The second protocol is designed for the case where an adaptive adversary may destroy communication going out of servers chosen according to the traffic observed by him. In this case a single message flows in a stream of K onions – the main point is that even when the adversary kills some of these onions, the stream quickly recovers to the original bandwidth – again K onions with this message would flow through the network.

Keywords: Anonymity, onion protocol, adaptive adversary

1. Introduction

Protocols for anonymous communication in computer networks attracted a lot of interest. Their importance increases together with growth of the threats in public networks. Many solutions were proposed, such as Chaum's DC-Nets (Chaum, 1988) and many variations of MIXes (Chaum, 1981). DC-nets pro-

*Partially supported by KBN scientific project 2003–2005 – grant number 0 T00A 003 23

vide information-theoretic security, but computational overhead is very high. For this reason, this solution is not regarded as practical for large scale applications. The second major proposal are Onions described in (Rackoff, 1993) for the very first time. In fact, it is based on idea of MIXes introduced in (Chaum, 1981). Anonymous communication based on onions is scalable and in certain scenarios meets very high demands on privacy (in other scenarios it provides essentially no protection). In order to provide anonymity, a message is sent not directly from the source to the destination, but through a path of randomly chosen nodes, where each node recodes the message with cryptographic tools, so that one cannot see any relationship between different versions of the same message. This protocol has many possible variants, see for instance (Freedman, 2002). Onions are the crucial component of Onion Routing (see for instance (Syverson, 1998) as a starting reference point).

1.1 Provable Security of Onion Routing

In certain scenarios one can really *prove* that onion protocol is secure, even if the adversary traces all traffic. The first rigid mathematical analysis was provided in (Rackoff, 1993). However, the authors assume that a large number of onions are sent at the same time and that the choice of intermediate nodes is somewhat restricted. The result, very interesting from theoretical point of view, is not sufficient for practical applications – security is guaranteed only for the onion paths which have a length that is polylogarithmic in the number n of servers (with a two-digit exponent). The last problem can be avoided by using another estimation (Czumaj, 1999) – the path length can be reduced to $O(\log^2 n)$.

A major breakthrough has been achieved by the change of adversary model in (Berman, 2004) – it is no longer assumed that the adversary can see all the traffic, but only a certain fraction of it. Even if some preferences of the users are known to the adversary, it is shown that the onion protocol does not reveal information through traffic analysis. Neither assumptions about the number of onions nor special addressing limitations are necessary. The path length required is a small degree polynomial in $\log n$. Finally, we have proved (Gomułkiewicz, 2004) that a path length of $O(\log n)$ is sufficient (which is optimal).

1.2 Drawbacks of Onion Routing

A systematic overview of adversary scenarios and their capabilities in real live situations was presented in (Syverson, 2000). The security proofs, mentioned above, should not give us any illusions that the onion protocol is secure in all circumstances. There is a number of tricks that can be used here, based

on the fact that connections are not static, exist over a certain time, and that the users have a certain behavior.

A timing attack exploits the fact that closing (resp. opening) a connection causes disappearing (resp. emerging) of one link both at the source and the destination. Monitoring these two hosts reveals immediately that the connection has closed (opened) without any complicated traffic analysis. A predecessor attack (Wright, 2003) is a refinement of this technique. An intersection attack (Berthold, 2000) may occur for instance when a user fetches a certain Web page (in an anonymous way) every time he starts a browser. An adversary records the users that are active at the time when this page is requested. The user in question appears quite often in these records... New attacks, also sophisticated ones, may emerge.

1.3 New Results

In this paper we propose how to deal with two problems. The first problem are node failures in the network. If a path of an onion goes through a node that is down, the message encoded inside the onion cannot be delivered. This is a consequence of the fact that private keys of the node that is down must be used to decode the message and to find out the next node on the path. In Section 3.1 we show that this is not a serious problem, since at each level we can encode alternative nodes through which the onion can be processed. Last but not least, this protocol is as secure as the original protocol in a passive adversary scenario considered in (Berman, 2004).

The second problem considered here is an adversary who can eavesdrop a certain fraction of the communication lines at each step; based on this information he may destroy all messages sent by arbitrarily chosen servers at the next step (however, the number of such servers is bounded). Of course, the original onion protocol is in a hopeless situation against such an adversary: he simply kills the onions one by one (not caring about their contents and destinations). In Section 3.2 we show how to cope with this problem. We propose a protocol such that K onions encoding a message m travel in parallel through the network. A major point in the construction is a mechanism that enables the stream of K parallel onions to self-recover, even if the adversary succeeds in killing all but one onion transporting m. The recovery mechanism must be not too aggressive, since the traffic induced may reveal to the adversary the points where the same message m is located. Then the recovery would bring more harm than profit: the adversary could destroy all messages transporting m. For this reason we propose a method that uses sparse communication, which is harder to be detected. In Section 3.2 we discuss shortly graph theoretic motivation of our solution.

2. Onion Protocol and Anonymity

2.1 Classical Onions

We consider a network with n servers, where each pair of servers may communicate directly. Each server has a pair of a public and a private key, all public keys are widely accessible.

Let us recall the onion protocol in one of the simplest versions. Assume that a message m has to be sent from node A to node B. For this purpose node A chooses at random λ intermediate nodes, say, $J_1, \ldots, J_\lambda$ (they need not to be distinct) and random strings $r_1, r_2, \ldots, r_{\lambda+1}$. Then A builds an *onion* $\mathcal{O}$ encoding m using the following recursive formula (Enc_X means encryption with the public key of X):

$$\begin{aligned} \mathcal{O}_\lambda &= \mathrm{Enc}_B(m, r_{\lambda+1}) \\ \mathcal{O}_i &= \mathrm{Enc}_{J_i}(J_{i+1}, \mathcal{O}_{i+1}, r_{i+1}) \quad \text{for } i < \lambda \\ \mathcal{O} &= \mathcal{O}_1 \end{aligned}$$

Then $\mathcal{O}$ is sent by A to J_1. Node J_1 decrypts the message with its private key. The plaintext obtained contains J_2, the name of the next server on the path, and $\mathcal{O}_2$ – the message to be sent to J_2. This is like *peeling off* the onion $\mathcal{O}_1$: we remove the out-most layer and forward the subonion obtained to the next server. This process of peeling off is repeated at each subsequent server until B gets finally the message m.

The idea behind is that each server J_i cannot see what is the contents of the subonion it sends to the next node – decryption of $\mathcal{O}_{i+1}$ requires knowledge of the appropriate private keys. So J_i cannot see the destination of the message for the subonion it possesses. However, note that additional measures are necessary to protect anonymity of communication. For instance, without the random strings r_i the following simple attack could be carried out: an adversary traces outgoing communication from J_{i+1}. When he detects a message Z sent from J_{i+1} to server U he checks whether $\mathcal{O}_{i+1} = \mathrm{Enc}_U(U, Z)$. If it is so, then $U = J_{i+2}$. This test can be carried out for each single step, so finally the adversary could detect the destination of the message encoded in $\mathcal{O}_i$ without breaking encryption scheme used.

In fact additional measures are necessary. For instance, the size of the packets sent could betray the path along which m is sent. So the encoding must be combined with appropriate padding (Chaum, 1981).

2.2 Adversary Models

There are many different models for an adversary who tries to break the onion scheme. This is a major issue, since a protocol resilient to attacks in one model might be vulnerable in another one. Also, too strong and unrealistic

assumptions about the adversary may lead to difficulties in showing security relevance of a protocol.

A passive adversary. A passive adversary may only eavesdrop messages transported along the network. We assume that the cryptographic encoding is strong enough, so the only information available is *where and when the messages have been sent.* It is often assumed that additionally an adversary may get information from a constant fraction of the servers.

There are few variants of the passive adversary:

1 Rackoff-Simon model: all communication lines can be traced by an adversary (Rackoff, 1993)

2 Berman–Fiat–Ta-Shma model: only a constant fraction of communication lines can be traced (Berman, 2004); these lines are determined in advance (with possibility that at each step a different set of lines is tapped),

3 the same as above, but the adversary can adaptively change his choice based on the traffic observed till this moment.

The second case is that an adversary is *active* and can get control over some number of servers. In this case the adversary may detour a subonion: instead of sending it directly to the next node J_{i+1}, a malicious server can encapsulate it with additional layers and send to J_{i+1} through a path of additional servers. This kind of attack can be traced by attaching some encoded confirmation that can be checked by the recipient of the message. Another idea is to send again the same subonion and trace where we can see subonions that have already appeared in the network. Repetitions reveal a path of the message traced. Through careful monitoring all the traffic such an attack can be detected, but it is unrealistic that the routers store and check all messages processed. As a defense one can use time stamps inside the packet (Kesdogan, 1998) – the subonions that do not arrive in predicted interval of time are immediately rejected. However, even with this approach not all problems are solved. A malicious server can postpone for a short moment all incoming traffic except one message. In place of the postponed onions it sends his own bogus messages with known routes ($(n-1)-$ *attack* from (Kesdogan, 1998)). Then analyzing the traffic is much easier: many routes are known by the adversary.

We consider the model of an adversary who removes the packets (either due to faults or with the aim to bring chaos into communication). This is a great problem: in the network with $n/\log n$ malicious servers and $\lambda = \log n$ each packet gets killed with probability $\Omega(1)$.

2.3 Vertex mixing vs. layer mixing

The adversary model has a big impact on the anonymity mechanism. The original idea of Chaum is that when at the same time two or more onions get

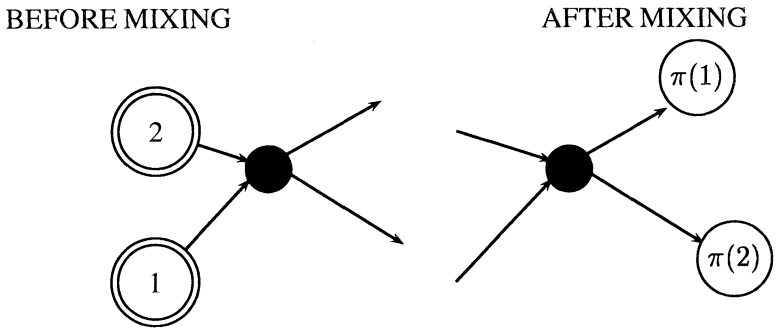


Figure 1. Vertex mixing: what an adversary can see.

into the same server that is not under the adversary's control, then this node acts as a mix: no relationship between incoming and outgoing onions can be found by an external observer. The problem is that if number of onions is moderate the chances that a given onion meets another onion are small. So a large number of onions is necessary to hide their routes.

In (Berman, 2004) Berman et al. pointed to this weakness and introduced "layer mixing" (to distinguish it from the "vertex mixing" discussed above). It is based on the assumption that an adversary may eavesdrop only a constant fraction of all communication lines. Then some number of onions are processed through hidden communication lines. Even if the adversary knows which nodes have received these messages, they are perfectly mixed. So the probability of mixing the messages within a layer of the protocol gets substantially larger.

3. *K*-Onion and Hydra-Onion Protocols

In this section we present modifications of the onion protocol that are aimed to make it robust against communication failures. Two kinds of failures are considered. Either the faults occur at random, or an adversary observes the traffic and tries to hit vulnerable points.

A general idea is that alternative routes for an onion are provided. At each hop, there is not one but at least two servers that may process the onion. So if one destination fails, the message can be sent to another destination.

3.1 DUO-Onions for Random Server Faults

3.1.1 Protocol Description. First we are concerned with random communication failures. All participants use a symmetric encryption scheme SEn. Let SEn_k stand for symmetric encryption with key k and Enc_X for encryption under public key of X.

The simplest version of the DUO-Onion protocol looks as follows: in order to send a message m to node B, node A chooses at random λ intermediate pairs of nodes, say, $(J_{1,1}, J_{1,2}), \ldots, (J_{\lambda,1}, J_{\lambda,2})$, random strings $r_1, \ldots, r_{\lambda+1}$, and keys for symmetric encryption $k_1, \ldots, k_\lambda$.

For each i, we demand that $J_{i,1} \neq J_{i,2}$, but the same server may be chosen to more than one pair. The onion $\mathcal{DO}$ is built via the following recursive formula.

$$\begin{aligned}
\mathcal{DO}_\lambda &= \big(\mathrm{Enc}_B(k_{\lambda+1}), \mathrm{SEn}_{k_{\lambda+1}}(m, r_{\lambda+1})\big)\ , \\
\mathcal{DO}_i &= \big(\mathrm{Enc}_{J_{i,1}}(k_{i+1}, 1), \mathrm{Enc}_{J_{i,2}}(k_{i+1}, 2), \\
&\qquad \mathrm{SEn}_{k_{i+1}}(J_{i+1,1}, J_{i+1,2}, \mathcal{DO}_{i+1}, r_{i+1})\big) \quad \text{for } i < \lambda\ , \\
\mathcal{DO} &= \mathcal{DO}_1\ .
\end{aligned}$$

The onions are processed in the following way: at a stage i, either $J_{i,1}$ or $J_{i,2}$ has $\mathcal{DO}_i$. First, using its private key, it retrieves k_{i+1} either from $\mathrm{Enc}_{J_{i,1}}(k_{i+1})$ or from $\mathrm{Enc}_{J_{i,2}}(k_{i+1})$. Then, with k_{i+1} it deciphers the third part of $\mathcal{DO}_i$, getting $J_{i+1,1}$, $J_{i+1,2}$, $\mathcal{DO}_{i+1}$. Then it tries to contact $J_{i+1,1}$. If it is down, it contacts $J_{i+1,2}$. Then it sends $\mathcal{DO}_{i+1}$ to the server which has responded. If no server responds, the transmission of the onion dies.

K-Onions Protocol works analogously – we have K possible destinations during a single hop instead of two.

A certain drawback is that each K-subonion contains K ciphertexts of the same symmetric key. However, this is not a problem since we expect to use only small values of K.

3.1.2 Delivery Probability. In this section we check that K-Onions are more efficient than just sending the same messages for many times.

Let us assume that r out of n servers executing the onion protocol are down. Since the onion paths are chosen at random, the location of these servers does not matter. Let $P(\lambda, k)$ denote the probability that a random k-onion reaches its destination server.

For the case of the classical onion protocol we get

$$P(\lambda, 1) = \left(\tfrac{n-r}{n}\right)^\lambda = \left(1 - \tfrac{r}{n}\right)^\lambda\ ,$$

where λ denotes the length of the onion path. Choosing $\lambda = c \log n$ (which is a secure length for an adversary eavesdropping a constant fraction of communication (Berman, 2004)), we get $P(\lambda, 1) = \Omega(1)$ for $r = O(n/\log n)$.

For k-onions we get:

$$P(\lambda, k) = \left(\frac{\binom{n}{k} - \binom{r}{k}}{\binom{n}{k}}\right)^\lambda = \left(1 - \frac{r}{n}\frac{r-1}{n-1}\cdots\frac{r-k+1}{n-k+1}\right)^\lambda \approx \left(1 - \left(\frac{r}{n}\right)^k\right)^\lambda,$$

For $\lambda = c \log n$, we get $P(\lambda, k) = \Omega(1)$ for $r = O(n/\sqrt[k]{\log n})$. For practical values of n, we may assume something like $\log n < 30$, so for $k = 5$ we get $\sqrt[k]{\log n} < 2$.

On the other hand, it is worth to say that it does not make sense to take large k, since the ratio between $P(\lambda, k)$ and $P(\lambda, 1)$ grows, but the rate of growth goes down. The biggest change occur for $k = 2$ and $k = 1$. Namely,

$$\begin{aligned}\frac{P(\lambda,k)}{P(\lambda,1)} &\approx \left(\frac{1-(r/n)^k}{1-r/n}\right)^\lambda \\ &= \left(\frac{(1-r/n)(1+r/n+(r/n)^2+\ldots+(r/n)^{k-1})}{1-r/n}\right)^\lambda \\ &= (1+r/n+(r/n)^2+\ldots+(r/n)^{k-1})^\lambda \approx (1+r/n)^\lambda\,.\end{aligned}$$

Practical example. Let us consider a network consisting of n servers where the number of faulty servers equals $r = 0.3n$ and the path's length $\lambda = 3$. In this case the usual onion reaches its destination with probability $P(3,1) \approx 0.34$ while $P(3,2) \approx 0.75$ and $P(3,3) \approx 0.92$.

3.1.3 Anonymity Issues.

Unlinkability. For an external adversary analyzing the traffic the k-onion protocol behaves just as the original onion protocol. So the results from (Berman, 2004) *do apply*.

On the other hand, the naive solution of sending the same message using multiple onions going through different routes may lead to weakening anonymity – traffic analysis might be facilitated by the fact that for each pair (sender, destination) there is a prescribed number of paths.

Adaptive attacks. A malicious server $J_{i,a}$ may send the subonion $\mathcal{DO}_{i+1}$ to $J_{i+1,2}$ instead of $J_{i+1,1}$, if $J_{i+1,2}$ collaborates with $J_{i,a}$. This enables them to reduce a little bit the unknown parts of the onion paths. By using similar arguments as in (Gomułkiewicz, 2004), one can show that it has the same effect as increasing the number of malicious servers by a constant factor.

On the other hand, a malicious server may start a small repetitive attack: it sends $\mathcal{DO}_{i+1}$ both to $J_{i+1,1}$ and $J_{i+1,2}$. Then both of them send a message to the same server – in this way the adversary may identify the server used on step $i+2$. On the other hand, the attack and the malicious server would be detected easily, so the attack is not attractive for the adversary.

Note that this trick does not reveal any useful information to the adversary – the only case in which the adversary obtains additional knowledge is when $J_{i+1,1}$ is malicious and pretends that it is down. The only knowledge he may gain by tracing communication sent by $J_{i,a}$ is the name of $J_{i+1,2}$. This knowl-

edge is of no advantage for him, since he may get the names of $J_{i+2,1}$ and $J_{i+2,2}$ by executing the protocol without any tricks.

3.2 Hydra-Onions – Fighting against Active Adversaries

Now we assume that the adversary traces a constant fraction of all communication lines and once it identifies the servers holding the same message, it blocks the outgoing communication from these servers. Of course, it is necessary that a message is transmitted via many routes – otherwise the adversary would win by simply killing the messages one by one.

The general idea is that we send a stream of messages encoding the same m. At each moment we have k subonions corresponding to m (provided that the adversary has not succeeded to kill some of them). Since the adversary may kill some of the subonions, we propose a mechanism that enables the stream to regenerate quickly, so that again we a have k subonions corresponding to m.

The construction must be careful, since a stream of messages encoding the same message may facilitate traffic analysis.

3.2.1 High Level Protocol Description. Assume that A has to send a message m to B. Then K intermediate nodes $J_{i,1}, J_{i,2}, \ldots, J_{i,K}$ are chosen by A for each $i \leq \lambda$. The main change to the previous protocol is that each of the servers $J_{i,1}, J_{i,2}, \ldots, J_{i,K}$ sends the onion to two servers from the list $J_{i+1,1}, \ldots, J_{i+1,K}$. Namely, $J_{i,j}$ sends a subonion to $J_{i+1,j}$ and to a randomly chosen server $J_{i+1,a(j)}$ where $a(j) \neq j$. The choice of $a(j)$ is made by A during onion construction.

In this way we achieve the following goals:

- since random bipartite graphs have expansion properties, if only a fraction of servers $J_{i,1}, J_{i,2}, \ldots, J_{i,K}$ received the subonion encoding m, after step $i+1$ the fraction of servers $J_{i+1,1}, J_{i+1,2}, \ldots, J_{i+1,K}$ holding a subonion encoding m increases with high probability;

- sending a copy of the subonion from each $J_{i,j}$ to all servers $J_{i+1,1}$,$J_{i+1,2}$, $\ldots$, $J_{i+1,K}$ would guarantee immediate recovery of the whole stream of K copies of subonions containing m. However, the communication pattern could betray that certain servers are holding a subonion corresponding to the same message. This could make killing m much easier. A sparse communication pattern proposed does not reveal such information to the adversary.

3.2.2 Protocol Description. For the sake of simplicity we describe and discuss the protocol for $K = 3$. Server A builds an onion $\mathcal{RO}$ via

the following recursive formula:

$$\mathcal{RO}_\lambda = \left(\mathrm{Enc}_B(k_{\lambda+1}), \mathrm{SEn}_{k_{\lambda+1}}(m, r_{\lambda+1})\right)$$

$$\begin{aligned}\mathcal{RO}_i = \Big(&\mathrm{Enc}_{J_{i,1}}(k_{i+1,1}, r_{i+1,1}), \mathrm{Enc}_{J_{i,2}}(k_{i+1,2}, r_{i+1,2}), \mathrm{Enc}_{J_{i,3}}(k_{i+1,3}, r_{i+1,3}),\\ &\mathrm{SEn}_{k_{i+1,1}}(J_{i+1,1}, J_{i+1,a(1)}, k'_{i+1}),\\ &\mathrm{SEn}_{k_{i+1,2}}(J_{i+1,2}, J_{i+1,a(2)}, k'_{i+1}),\\ &\mathrm{SEn}_{k_{i+1,3}}(J_{i+1,3}, J_{i+1,a(3)}, k'_{i+1}),\\ &\mathrm{SEn}_{k'_{i+1}}(\mathcal{RO}_{i+1})\Big) \quad \text{for } i < \lambda\end{aligned}$$

$$\mathcal{RO} = \mathcal{RO}_1$$

In this protocol it is not the case that subonions are sent. Namely, when a server J has to send $\mathcal{RO}_i$ to server J', then $\mathcal{RO}_i$ is encrypted together with a random nonce with a public key of J' before it is sent to J'. Alternatively, we may use a probabilistic asymmetric encryption scheme (such as ElGamal) for encapsulating $\mathcal{RO}_i$.

Let us describe how a subonion is processed. Assume that J receives an (encapsulated) subonion $\mathcal{RO}_i$. Then J decodes $\mathcal{RO}_i$ and deciphers the first three components of $\mathcal{RO}_i$ with the private key of J. In this way, J obtains three symmetric keys k, k', k''. Then J deciphers the 4th, the 5th and the 6th components of $\mathcal{RO}_i$ with the keys, respectively, k, k', and k''. In one case, J obtains the valid key k'_{i+1} and the names of two servers J', J'' for the next hop. (If necessary, we may include some characteristic string in the plaintext in order to detect easily which of the keys k, k', and k'' is valid.) Having k'_{i+1}, server J deciphers the last component of $\mathcal{RO}_i$ and retrieves $\mathcal{RO}_{i+1}$. Then J encapsulates $\mathcal{RO}_{i+1}$ as described above and sends the results to the servers J' and J'', respectively.

3.2.3 Recovery Properties. Assume that the adversary is not blocking the communication at the moment and there is only one server with a subonion holding m. Then after one step we get 2 servers with a subonion holding m with probability 1. If there are already two servers keeping subonions with m, then after one step we have still 2 such servers with probability $\frac{1}{4}$ and 3 such servers with probability $\frac{3}{4}$. Since the numbers $a(j)$ are chosen independently at random, our experiment corresponds to Bernoulli trials with success probability $\frac{3}{4}$. So there is no success within t trials (meaning that we have still only 2 servers holding m) with probability $\frac{1}{2^{2t}}$.

For the case when $K > 3$ the arguments are more tedious. However, let us point that the following Markov chain $\mathcal{S}$ converges quickly. The states of $\mathcal{S}$ are nonempty subsets of $\{1, \ldots, K\}$. The transition function of $\mathcal{S}$ can be described as follows: let U be the current state of $\mathcal{S}$; then for each $a \in U$

choose independently at random an element $r(a) \in \{1, \ldots, K\} \setminus \{a\}$. Then the new state of $\mathcal{S}$ is the set $U \cup \{r(a) | a \in U\}$. Due to expansion properties of random graphs, the chain $\mathcal{S}$ converges quickly to the state $\{1, \ldots, K\}$. We skip further discussion on this problem, since we think that small values of K are most important and for these values the convergence rate can be easily estimated.

3.2.4 Resilience to Attacks. As mentioned, an adversary who analyzes the traffic at step i may locate the servers holding the same message m and kill all packets sent from servers holding m at step $i + 1$. In this way m would disappear, even if the adversary does not know the contents and the destination of m.

We assume, as in (Berman, 2004), that the adversary may eavesdrop only a constant fraction of communication lines. Recall (Berman, 2004) that servers $J_1, \ldots, J_m$ and $J'_1, \ldots, J'_m$ form a *crossover structure*, if no communication line (J_a, J'_b) for $a, b < m$ is eavesdropped by the adversary. The number m will be called *crossover size*.

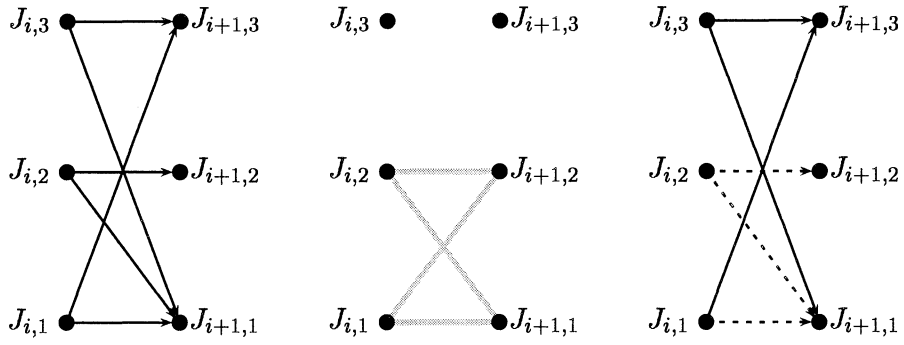
The ideal situation is when the servers $J_{i,1}, \ldots, J_{i,K}$ and $J_{i+1,1}, \ldots, J_{i+1,K}$ from the definition of an onion $\mathcal{RO}$ form a crossover structure at step i. Then the adversary has no trace that they belong together to a certain message m. What is the probability that such a case occurs? It is hard to answer this question: the adversary may adopt some clever strategy to choose the links eavesdropped so that as few as possible crossover structures occur. It turns out that there are graph theoretical limitations on the adversary. Noga Alon (Alon, 2001) (Corollary 2.1) shows the following result:

LEMMA 1 *For every fixed $\epsilon > 0$, and every fixed integer $t > 0$, and for any graph G with n vertices and at least ϵn^2 edges, the number of subgraphs of G isomorphic to $K_{t,t}$ (bipartite complete graph with t vertices on each side) is at least:*

$$\frac{1}{2}\binom{n}{t}\binom{n}{t}(2\epsilon)^{t^2}.$$

In our case we say that an edge (J, J') belongs to G, if the communication link between j and J' is not eavesdropped by the adversary. The main point in Lemma 1 is that a lower bound on the number of crossover structures does not depend on the structure of G, that is, on the strategy of the adversary.

In order to examine the case $K = 3$, let us note the following: the probability that $J_{i,1}, J_{i,2}, J_{i,3}$ and $J_{i+1,1}, J_{i+1,2}, J_{i+1,3}$ form a crossover is at least $f^9/2^{10}$ for a fraction f of links that are not under the adversary's control. In this case the probability that the adversary kills all packets holding the message processed by these servers is about $(r/n)^3$, where r is the number of servers that can be blocked at step $i + 1$.



(*a*) communication pattern (*b*) crossover (*c*) adversary's view

Figure 2. Crossovers of size 2 and the viewpoint of the adversary.

The adversary may be in trouble even if no crossover of size 3 occurs. With probability at least $f^4/2^8$ a crossover of size 2 is formed by two of the servers from $J_{i,1}, J_{i,2}, J_{i,3}$, say J_{i,x_1}, J_{i,x_2} and by two of the servers from J_{i+1,x_1}, J_{i+1,x_2}. By a simple case inspection (see also Fig. 2) we can check the following fact:

LEMMA 2 *Assume that a crossover of size 2 is formed by two servers J_{i,x_1}, J_{i,x_2} from the list $J_{i,1}, J_{i,2}, J_{i,3}$ and by two of J_{i+1,x_1}, J_{i+1,x_2}. Then for every choice of $a(1), a(2), a(3)$, the communication lines between $J_{i,1}, J_{i,2}, J_{i,3}$ and $J_{i+1,1}, J_{i+1,2}, J_{i+1,3}$ where the adversary observes a traffic corresponding to message m does not form a connected graph.*

The meaning of Lemma 2 is that when a crossover of size 2 occurs, then the adversary cannot link together the nodes $J_{i+1,1}, J_{i+1,2}, J_{i+1,3}$ – at least one of then is an *orphan*. Since we expect that there are many such orphans the adversary is in trouble: killing m will succeed only if all three servers $J_{i+1,1}, J_{i+1,2}, J_{i+1,3}$ are blocked.

For larger parameters K, the adversary should be even more confused (we postpone the analysis to the full version of the paper). Let us explain some intuitions behind. For the moment even assume that the adversary knows the relationships of the kind $J_{i,j} - J_{i+1,j}$ and that these relationships are the same for each onion. We draw a directed *additional link multi-graph* $\mathcal{A}$ describing step i. It has n vertices, with vertex j representing the jth server. We draw an arc $\vec{JF}$ in $\mathcal{A}$ when at step i server J sends a message to F, where F is the "second location" indicated by function a. Let us consider the arcs related to

processing the same onion. Since each node has out-degree at most 1, we get a subgraph of $\mathcal{A}$ with some specific properties (see Fig. 3): there are exactly K arcs, each connected component contains a single circle and some number of directed paths leading to the circle. The unlucky case for the adversary is when there is more than one connected component in this graph – the adversary cannot link the servers appointed to the same message provided that there many other components due to other messages. It is known that for large K the size of largest connected component divided by K is a random variable with probability distribution that converges with K to a Poisson-Dirichlet distribution.

However, even if there is exactly one component, the adversary might be in trouble, since we assume that the adversary does not eavesdrop all communication links, but only a certain fraction of them. In this case the graphs such as depicted in Fig. 3 get disconnected, since the adversary does not know the status of a constant fraction of communication lines (see Fig. 4).

4. Conclusions and Open Problems

We disregard the problem of different degrees of vulnerability of the servers and communications lines. However we should be able to assign different elements of the network different probabilities of failure or corruption by an adversary. It is yet unclear how to adopt the onion protocols to this situation.

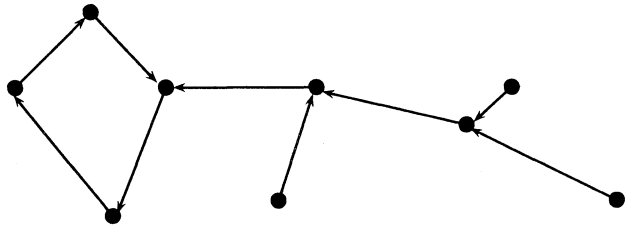


Figure 3. Example arcs in $\mathcal{A}$ corresponding to the same message.

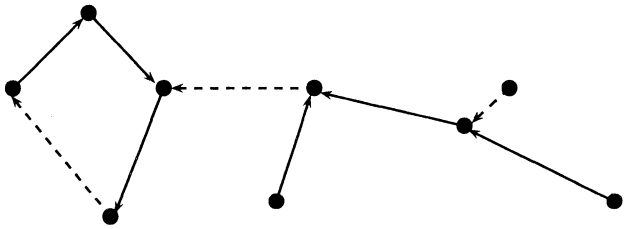


Figure 4. Adversary point of view for the situation depicted by Fig. 3.

References

Alon, N. : Testing Subgraphs in Large Graphs. ACM-SIAM FOCS 2001, 434-439.

Berman R., Fiat A., Ta-Shma A.: Provable Unlinkability Against Traffic Analysis. Accepted for Financial Cryptography 2004.

Berthold, O., Federrath, H., Köhntopp, M.: Project "Anonymity and Unobservability in the Internet." Workshop on Freedom and Privacy by Design / CFP2000, ACM, 2000, 57-65.

Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. CACM 24(2) (1981) 84-88.

Chaum, D.: The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. Journal of Cryptology 1(1) (1988), 65-75.

Czumaj, A., Kanarek, P., Kutyłowski, M., Loryś K.: Distributed Stochastic Processes for Generating Random Permutations. 10 ACM-SIAM SODA, 1999 271-280.

Freedman, J., Sit, E., Cates, J., Morris, R.: Introducing Tarzan, a Peer-to-Peer Anonymizing Network Layer 1st International Workshop on Peer-to-Peer Systems (IPTPS02), Lecture Notes in Computer Science 2429. Springer-Verlag, 2002, 121-129.

Gogolewski, M., Kutyłowski, M., Łuczak, T.: Distributed Time stamping with Boomerang Onions. Manuscript.

Gomułkiewicz, M., Klonowski, M., Kutyłowski, M.: Provable Unlinkability Against Traffic Analysis already after $\mathcal{O}(\log(n))$ steps!. Manuscript, 2004.

Kesdogan D., Egner J., Büschkes R.: Stop-and-Go-MIXes Providing Probabilistic Anonymity in an Open System. Information Hiding '98 Lecture Notes in Computer Science 1525. Springer-Verlag, 83-98.

Syverson P. F., Reed M. G., Goldschlag D. M.: Private Web Browsing. Journal of Computer Security Special Issue on Web Security 5 (1997) 237-248.

Syverson P. F., Reed M. G., Goldschlag D. M.: Anonymous Connections and Onion Routing. IEEE Journal on Selected Areas in Communication. 16(4) (1998) 482-494.

Syverson, P., Tsudik, G., Reed, M., Landwehr., C.: Towards an Analysis of Onion Routing Security. Workshop on Design Issues in Anonymity and Unobservability, July 2000.

Rackoff C., Simon D.R.: Cryptographic Defense Against Traffic Analysis. 25 ACM Symposium on Theory of Computing (1993) 672-681.

Wright, M., Adler, M., Levine, B., Schields, C.: Defending Anonymous Communication Against Passive Logging Attacks. IEEE Symposium on Security and Privacy 2003, IEEE Computer Society, 28-38.