

# How to Achieve Blocking Resistance for Existing Systems Enabling Anonymous Web Surfing

Stefan Köpsell  
Institute for System Architecture  
Dresden University of Technology  
01062 Dresden, Germany  
sk13@inf.tu-dresden.de

Ulf Hillig  
ulf.hillig@blackbrain.de

## ABSTRACT

We are developing a blocking resistant, practical and usable system for anonymous web surfing. This means, the system tries to provide as much reachability and availability as possible, even to users in countries where the free flow of information is legally, organizationally and physically restricted. The proposed solution is an add-on to existing anonymity systems. First we give a classification of blocking criteria and some general countermeasures. Using these techniques, we outline a concrete design, which is based on the JAP-Web Mixes (aka AN.ON).

## Categories and Subject Descriptors

H.3.5 [Online Information Services]: Web-based services

## General Terms

Design, Experimentation

## Keywords

JAP, Mix, AN.ON, Blocking Resistance

## 1. INTRODUCTION

“Everyone has the right to freedom of opinion and expression; this right includes freedom to hold opinions without interference and to seek, receive and impart information and ideas through any media and regardless of frontiers.” (Article 19, Universal Declaration of Human Rights)

However in some countries of the world the free access to information (for instance) via Internet is restricted. In such countries a  *censor*  tries to hinder people from reading documents with special political, ethical, sexual or religious content. This is called  *censorship* . One can use  *censorship resistant publishing systems*  to enable people (blockees) to read even censored documents. Some examples of censorship resistant publishing systems are Freenet [10], Free Haven

[11], Publius [29], Tangler [28] and GNUnet [6]. Unfortunately all of them have the same disadvantage: the censor can easily  *block*  the access to the entire service. In a certain manner such a  *blocker*  does an attack against the availability and reachability of that service. Censorship and blocking are related in that sense, that blocking can be used as a kind of undirected (or unspecific) censorship.

Much information about censorship in different parts of the world can be found on the Internet and in newspapers, e.g. in [32] Jonathan Zittrain and Benjamin Edelman from Berkman Center for Internet & Society at Harvard Law School present their research results about Internet filtering practices in different countries and organizations.

After having developed the JAP-Web Mixes anonymizing service [7] and making it public, our experiences is that (i) our service was used by people with restricted Internet access simply to get information which they otherwise could not get and that (ii) the censors of countries these people live in block the access to the anonymizing service. For these reasons our goal is to develop a practical, usable and blocking resistant system for  *anonymous*  web surfing. We strongly believe that an information retrieval system which bypasses censorship has to guarantee the anonymity of its users, because often the consumption of “forbidden” (censored) information is illegal.

We are aware of the fact that against a blocker with enough power (in terms of resources) probably no blocking resistant system will exist. Nevertheless we will implement our system and bring it to the public to get a “proof of concept”. Besides, another goal is to learn as much as possible about the behavior (strategies) and capabilities of a blocker. In order to achieve this we will not implement the strongest possible system we have in mind but release our ideas step by step. In this way we will see which building blocks of our blocking resistant system are really necessary and in which way and how fast the blocker will adapt his blocking strategies.

The rest of this paper is structured as follows. In Section 2 we describe known solutions, which are related to our problem. In Section 3 we present general assumptions of the blocking scenario in terms of a threat model. Thereafter in Section 4, a classification of blocking criteria is given. Based on these criteria, Section 5 outlines some general techniques to bypass blocking, and Section 6 describes the actual design of our system. Finally, Section 7 contains our conclusions and open questions.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

WPES'04, October 28, 2004, Washington, DC, USA.  
Copyright 2004 ACM 1-58113-968-3/04/0010 ...\$5.00.

## 2. KNOWN SOLUTIONS

### 2.1 Classes of known solutions

We identify three classes of known solutions:

1. *Censorship resistant publishing systems* as mentioned above. These systems could be used to publish a document, which could not be censored by the blocker. One disadvantage is that often the protocols are designed so that a client can learn about all available servers, thus a blocker can use these mechanisms to automatically block them. Another disadvantage is that only documents within the publishing system can be accessed but normal web surfing is not possible.
2. *Anonymity systems* like Crowds [23], Onion Routing [18], MorphMix [25], Tarzan [17] or Tor [12] make censorship more difficult because communication relations are hidden from the blocker. Nevertheless these systems are not really designed to guarantee blocking resistance. A system either has some centralized parts (nodes) which could be blocked or information about all participating nodes could easily be collected and used to block access to the system.
3. *Blocking resistant systems*. Some of them are described in the following section. They offer some interesting components but lack of providing strong anonymity.

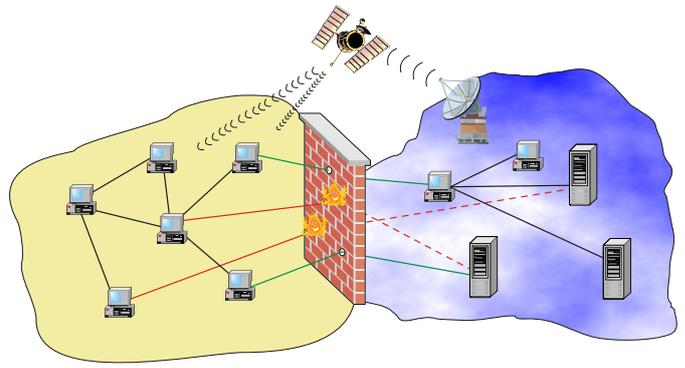
### 2.2 Known solutions for blocking resistance

All known systems share the same idea: try to establish as many access points (nodes) to the blocked service as possible. The hope is that the blocker is not able to block all these nodes. Some of the existing systems are described below to illustrate some aspects and enhancements of this general idea.

SafeWeb in conjunction with TriangleBoy was a solution offered by SafeWeb Inc. in order to "...build a network of servers for Chinese Web surfers to access sites censored by their government." [24] Technical SafeWeb is a simple anonymizing proxy which could be accessed using SSL. Therefore and because of some design flaws [20] the achieved level of anonymity is very low. TriangleBoy was an add-on program for SafeWeb which should be operated by volunteers. The goal was to establish a multi-node, one hop forwarding network. There is no additional, detailed information about certain design aspects available (for instance how the information about available TriangleBoy nodes should be distributed).

The Peekabooty Project [2] aims to establish a blocking resistant peer-to-peer network using similar techniques as TriangleBoy like SSL for accessing the nodes. They use multi hop (Crowds-like) forwarding to achieve anonymity. Some slides on their web page also mention steganography but Peekabooty is currently being redesigned and rewritten so neither detailed information about the protocol nor any software currently exists.

The Six/Four System [3] adds the idea of Trusted Peers, which act as exit nodes of the peer-to-peer network. All requests are encrypted using a public key of a trusted node, such that no man-in-the-middle knows the content of the request. As the word implies, the user has to trust a Trusted Peer, which is certified in some way.



**Figure 1: Blocking scenario.** The blockees are on the left side and the blocker interferes between them and the rest of the Internet.

A more sophisticated system is Infranet [13] which uses steganographic techniques to establish a hidden channel between a user and a forwarder. Such a node acts also as normal web server and the hidden channel is embedded into allowed HTTP traffic (for upstream communication the information is embedded into the requested URLs and for downstream the information is embedded into downloaded images). Anonymity was not a design goal and so is not provided. Another disadvantage is, that for a practical and usable system many operators of web servers have to install the Infranet add-on (which does not happen until now). Also no solution for distributing the information about available Infranet forwarders is given. Eventually the authors do not care much about robustness of steganography, so that the blocker can destroy the hidden channel (even if he does not know if one exists or not) by rewriting URLs or mounting StirMark<sup>1</sup> attacks against the downloaded images. Hence it is unclear if the bandwidth consuming steganography is really necessary to prevent blocking.

Keyspace Hopping [14] is a technique which tries to solve the problem of distributing the information about available forwarders. The goal is that no one gets the whole information about all forwarders. This ensures that a blocker could not block all of them. One assumption is that each of the censored users has a different IP address but if the censor uses Network Address Translation (NAT) techniques this assumption does not hold.

## 3. THREAT MODEL

Before we start with the discussion of our design decisions, we explain our assumptions about the world and especially about the capabilities of the blocker. These assumptions are similar to the ones proposed by Peekabooty [1]:

- *The blocker allows only partial access to the Internet, but for political or economical reasons he is not willing to prevent all Internet traffic.*
- *Some (small bandwidth) information flow into the country exists.* This information flow (e.g. satellite based broadcasting) cannot be controlled by the blocker and

<sup>1</sup>StirMark is a benchmark tool for robustness testing of image watermarking algorithms which uses for instance bilinear geometric distortions. These attacks could also be used to destroy steganographic content.

could be used for bootstrapping the blocking resistant system.

- “*Global thinking and local acting*” volunteers are willing to support. These people will give some of their resources (computing power, Internet bandwidth etc.) to support people with restricted Internet access.
- *The available resources differ among blockees.* Someone may have a high performance PC with high speed Internet access whereas others only have outdated computers with modem connections.

The attacker (or blocker) model we have in mind has the following properties. The blocker:

- *owns huge amounts of resources*, including money, computing power and human resources. We assume that human resources remain the most expensive ones and cannot be raised as easily as computing power.
- *controls all (network) links and nodes (routers, proxies etc.) to the outside world.* He can read and analyze all traffic; can delete, change and insert data. He, himself, has also free access to the Internet.
- *does not control (large parts of) the ‘outside’ Internet.*
- *knows everything about the design of the blocking resistant system* including how the system works and the reasons for every design decision. There is no “security by obscurity”.

## 4. CLASSIFICATION OF BLOCKING CRITERIA

In general we can classify blocking criteria depending on the involved *communication layer* (according to the ISO/OSI reference model or the TCP/IP protocol suite [26]) and if the characteristic is based on the *content of communication* or the *circumstances of communication* (sometimes it is difficult to decide something belongs to content or circumstances; especially because the content of a lower communication layer may describe the circumstances for a higher communication layer). Appendix A contains a schema of our classification.

All these characteristics could be used together with predicate logic to form very complex blocking strategies. Also the blocker can decide if he wants to model allowed or forbidden communication (blacklist, whitelist) and can mix these.

### 4.1 Blocking based on the circumstances of communication

There exist different kinds of general circumstances of communication which could be used for blocking decisions:

1. *Addresses*, that could be further divided into:
  - sender address (or source address)
  - receiver address (or destination address)
  - kind of address (unicast, multicast or broadcast)
  - place of physical presence of sender or receiver
2. *Timing*, including
  - time of sending

- time of receiving
- duration
- frequency

3. *Data Transfer*, which could be characterized by:

- circuit switched, packet switched or broadcast communication
- simplex, half- or full-duplex
- bandwidth, latency, amount
- connection-oriented or connectionless
- reliable or unreliable

4. *Services*, which could be characterized by:

- protocols
- names
- addresses

These characteristics could be more precise according to the involved communication layer. Next we give just some examples. On the lowest layer (physical/data link layer [OSI]; host-to-network layer [TCP/IP]) a blocker could forbid the use of telecommunication links (plain old telephone or GSM) for data transmission. The detection of data transmission even if done via analogue telephone lines is easy because the signaling is standardized by the ITU. Combining this with address information could lead to blocking the access to foreign ISPs via modem. On the network layer (or Internet layer) communication often is blocked depending on source or destination IP addresses. On the transport layer often port numbers are used to decide if the data transfer belongs to a certain application level protocol or service. This kind of blocking could be improved with the help of application level firewalls or protocol analyzers. Some web proxies restrict the maximum size a downloaded object may have to hinder people from downloading music or movies. Timing or duration related blocking may become a powerful tool for blockers to mount intersection attacks [8] and thus undermine the anonymity a blocking resistant system may offer.

### 4.2 Blocking based on the content of communication

Apart from the circumstances, a blocker could also look at the content of communication. As already mentioned it is a little bit difficult to distinguish between these two categories as “content” of a certain protocol layer may be address information of another layer. If we talk about content we mean transferred data in the sense of end-to-end communication (payload).

Again the blocking decision could depend on the kind (or type) of content and the content itself. If the content consists of files or objects (FTP, HTTP, e-mail) the type of that content could be identified simply by exploring file extensions or MIME<sup>2</sup> types. Improved procedures can analyze the file structure to find out if “mypicture.gif” is really an image. Statistical examinations can detect encrypted or compressed content.

<sup>2</sup> MIME: Multipurpose Internet Mail Extensions (RFC’s 2045 thru 2049) is a standard system for identifying the type of data contained in a file based on its extension.

Regardless of file- or stream-oriented communication (Chat, Voice over IP, Video Conference) the blocker can use pattern matching techniques to check the content itself. He can scan e-mails, newsgroups, web pages or chat rooms for forbidden words or phrases. In [19] it is described how “fingerprints” of web pages could be used to identify the download of a certain web page even if the communication is encrypted using SSL. The idea is, that downloading a web page together with its embedded objects forms a typical data stream (amount of data, number of requests). This “fingerprint” could be matched against reference values within a database.

## 5. COUNTERMEASURES AGAINST BLOCKING

### 5.1 General

We now describe general ideas or building blocks to bypass blocking. In section 6 we will describe our design in detail. Realizing blocking resistance can be divide in two tasks:

1. an infrastructure for accessing the anonymizing service itself
2. distribution of information about this infrastructure

Note that in contrast to the first task the second does neither need high bandwidth nor low latencies.

The common requirement for both is that the necessary data transmissions has to be unsusceptibly. Therefore one should use widely-used protocols like SSL, HTTP or SMTP etc. In [15] a transport layer abstraction for peer-to-peer networks is proposed and a SMTP implementation is described in detail. Especially SMTP seems to be useful because sociological research has shown that any person (known by name, address and profession) could be reached at an average of six times of e-mail forwarding [30]. Adopting this fact a blockee can send an e-mail to (a randomly chosen) person asking for help (for instance requesting information about an access point). Even if this person does not know anything about the blocking resistant system he perhaps forwards this mail which will hopefully reach a volunteer at the end. Of course this technique should only be used for bootstrapping (e.g. if a blockee has no other way to get information about available access points). Otherwise the potential volunteers become angry about all this “spam”. Nevertheless a big advantage is that no additional information about this kind of volunteers needs to be published making blocking more difficult.

*Steganography*<sup>3</sup> is one of the proposed techniques which could be applied to all communication layers. For example steganography in (digital) telephone calls [22], [16] could be use to access a foreign ISP whereas steganography in digital images or video conferences could establish a hidden TCP/IP connection. From a practical point of view, one problem is the (embedding) ratio between the amount of embedded data and the required amount of cover data. For a

<sup>3</sup>Steganography means the hiding of a secret message (signal) within an ordinary message (signal) and the extraction of it at its destination. Beside sender and receiver nobody must be able to reveal the existence of the secret message. Digital watermarking in contrast has the goal to embed some (small amount of) data so that it is impossible to remove it without destroying the cover (robustness of watermarking).

secure steganographic system this is at least 1:10 [31] meaning a download of 1 Mbyte web content needs 10 Mbyte of transmitted cover data. In addition a blocker (in contrast to usual steganalysis) does not need to break a given steganographic system in a way that he can decide for every image with high accuracy whether something is embedded or not. It is sufficient to get a suspicion because he can check if a node really offers steganographic communication simply by acting like a blockee. As countermeasure the used steganographic algorithm should be randomly chosen, since each steganographic algorithm has its own specific attacks. If the concrete algorithm is unknown to the blocker his expenditure of analysis raises and more false positives occur. As described above another problem is the robustness of steganography (see Section 2). As far as we know there is no secure steganographic system which is robust against active attacks. Nevertheless we expect that such a system would have a much worse embedding ratio (probably 1:100).

Whenever steganography (as well as cryptography) is used the algorithms should have an additional property: if a blockee is suspected for using steganography (cryptography) he should be able to prove that he only exchanges harmless messages (known as plausible deniability). In the context of steganography this could be done by embedding a second message which could be uncovered just in case. Of course this will destroy some part of the already embedded message but that is not problematic as the steganographic algorithm has to provide robustness in any case as mentioned above.

Using steganography is just one instance of the more general concept to force a blocker to block “*all or nothing*”. Encryption is another example for this. The idea is that the blocker cannot decide (based on his observations) if certain transmitted data belongs to forbidden content or not. If for example all e-mails have to be encrypted around the world then a blocker could not scan them for blacklisted words or phrases.

### 5.2 Blocking resistant infrastructure for anonymous communication

This section describes the main ideas concerning the infrastructure. In general it could be divided into two parts:

1. The anonymizing service itself. We assume that it will work as a “stand-alone” system, providing strong anonymity to its users. Although we know that such a system does not exist at the moment, we assume that it will exist in the future and do not discuss the realization of this part, because it is out of scope of this paper.
2. The blocking resistant infrastructure to access the anonymizing service. We understand this as an “add-on”.

A well know approach is the creation of “*many access points*”. The assumption is, that the blocker is not able to block all of them. Often such an access point is simply a forwarder. Normally the blockee has to connect to the forwarder, which then sends the data to the anon service. In this case we have to publish a list of participating volunteers and have to ensure that a blocker will not get full access to this list.

However it is also possible that a volunteer establishes a connection to a blockee on request of this blockee. In this case information about volunteers does not need to be

published. Of course now a blockee has to let a potential volunteer know that he needs his help. Naturally the best solution is to mix these two concepts.

Note that the opposite of “many access points” is possible, too: imagine that all web pages of the United States are only retrievable (from abroad) by sending encrypted request to one and only one special node.<sup>4</sup> Clearly this idea belongs to the “all or nothing” concept because a blocker has to block all requests to this node.

A problem of the “many access points” approach is, that the blocker itself could operate many forwarders and thus tries to attract as many blockees as possible. In order to make this attack more tricky, a blockee’s client should select the forwarder depending on some client and forwarder properties. Keyspace Hopping [14] is a technique, which tries to fulfill this by giving a blockee only partial information about available forwarders depending on the client’s IP address. Also, a client chooses the forwarder regarding its IP address. Of course this technique is only useful if the blocker can not easily fake these IP addresses by some kind of NAT technologies.

Therefore the distributed directory publishes information about a forwarder only if it contains a credential (e.g. a digital signature) given by a trustworthy introducing server. This server checks whether the IP address is not faked by trying to establish a connection to the forwarder. As we assume the blocker does not control the “free” Internet, thus he cannot reroute this traffic to its faked forwarders. The necessary information (e.g. a public key) to verify these credentials can either be distributed with the client software or is published using the small band communication into the country.

Additionally, the forwarder itself could show some credentials proving for instance his nationality. This functionality is provided by currently deployed technologies, such as digital signature cards or PGP’s web of trust. All this would become more easy if future technologies like identity management etc. are widely in use.

### 5.3 Distribution of information about the forwarders

How to distribute the information about available access points is still an open problem. In general it needs to fulfill two conditions:

1. The access to the information could not be blocked itself.
2. The information has to be fuzzy.

The first condition looks like introducing a recursive hen-egg problem but the requirements (bandwidth, latency etc.) of a service for surfing the web and a service for distributing information about access points are very different. Also we need this “out-of-band” information distribution only for bootstrapping. After the user has once got a successful connection to the blocking resistant system he can retrieve all necessary information “in-band” (within the system itself). For this reason many different solutions are thinkable for

<sup>4</sup>Of course the given example is just a simple one to illustrate the idea, because a blocker could enforce all users within his sphere of control to send unencrypted requests to him, for which he decides if he will encrypt and forward them to the special node.

bootstrapping: Broadcast via satellite or short wave; using e-mails in combination with steganography etc. As described in the general assumptions section (see above), we assume that some (small bandwidth) information flow into the country exists.

The second requirement arises from the fact, that we cannot distinguish a blockee from a blocker. If we would be able to distinguish them we could give information about access points only to blockees.

One concept is to let the client solve puzzles. The idea behind this is that a blockee only needs to solve *one* puzzle while a blocker has to solve *all* puzzles.<sup>5</sup>

One kind of such puzzles relies on cryptography[21]. Unfortunately cryptographic puzzles seem to be not feasible. The hardness of a puzzle has to be well dimensioned (in terms of computing power) so that a blocker needs plenty of time to solve all puzzles, but blockees with outdated computers do not.

To overcome this limitation puzzles which are independent from computing power are needed. The requirements for such puzzles are:

- most humans can solve the puzzles
- current computer programs cannot solve the puzzles
- given some information a related puzzle can be constructed automatically

The first requirement ensures that every blockee can solve the puzzle within nearly the same time especially independent of resources he owns. The second requirement ensures that a blocker really needs (expensive) human resources and cannot simply buy more computing power. The third requirement is necessary to make the system feasible, because no volunteer who operates a forwarder would be willing to create puzzles like picture puzzles or short stories by hand every time the IP address of his forwarder changes.

In [27] a cryptographic protocol (CAPTCHA<sup>6</sup>) is introduced “whose underlying hardness assumption is based on an AI problem.” The goal is to generate and grade test that: a) most humans can pass and b) current computer programs cannot pass (see Figure 2 for some examples). The proposed algorithms can be used to generate the puzzles we need. Also the authors claim that a CAPTCHA and all used data could be known to the attacker (this is the word “public” stands for in CAPTCHA). This perfectly meets our design requirements. We can use CAPTCHAs in two ways: for constructing the puzzles for the information itself and for constructing interactive protocols which ensure that really a human requests the information (puzzle) and not a computer.

Unfortunately there exists an attack on CAPTCHAs which the authors call “stealing cycles from humans”. Imagine that the blocker operates a porn web site. As soon as he discovers a CAPTCHA test he displays this puzzle on his porn web site granting access to the pictures only if the puzzle is solved. This way many “volunteers” from around the world will help the blocker. A countermeasure could be

<sup>5</sup>But we have to keep in mind that a blocker can gather address information also from other sources. He can for instance scan all incoming and outgoing traffic for suspicious patterns or circumstances.

<sup>6</sup>CAPTCHA: Completely Automated Public Turing test to tell Computers and Humans Apart



Gimpy algorithm: Try to read the words  
 Pix algorithm: What are these pictures of?

**Figure 2: Two examples of CAPTCHAs which are taken from The CAPTCHA Project (<http://www.captcha.net/>)**

to construct the puzzle so that only a small partition of the peoples is able to solve them. Of course the blockees have to belong to this part. Imagine that the blockees live in China. In this case we can use Chinese characters to create CAPTCHAs because only a few people from outside China would be able to understand them. Additional we should include blocked information itself. In the scenario above we could have slogans in the background like: “Demonstrate for a free Tibet” etc. Now the blocker cannot misuse “ordinary” Chinese people as volunteering CAPTCHA solvers.

Another way to bind as many human resources of the blocker as possible may be to “poison” the information about available access points. A poisoned list contains (beside the information of forwarders) for instance IP addresses of (web) servers which a blocker definitely does not want to block (for political or economical reasons). Therefore he cannot simply enter such a list into his firewalls.

## 6. DESIGN OF A BLOCKING RESISTANT SERVICE FOR ANONYMOUS WEB SURFING

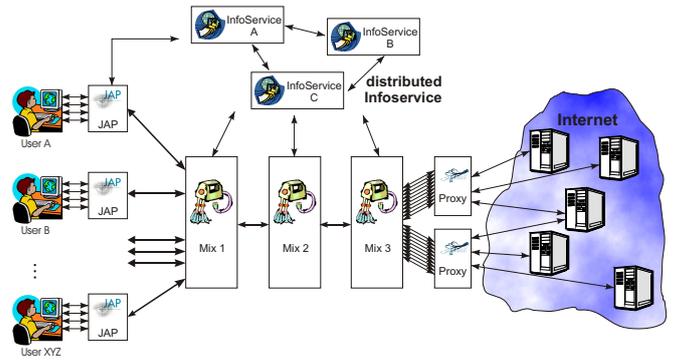
Our solution is an add-on to an existing anonymity service (namely AN.ON) and can be included in the already widely deployed client software (JAP), which is needed to access the anonymity service. This is a big advantage compared to other systems like Peekabooty, TriangleBoy or Infranet where the volunteers get no additional benefit.

We utilize the “many access points” idea, thus every JAP can act as a forwarder if the direct access to the anonymity service is blocked.

The next paragraph shortly describes the AN.ON system. Only the facts necessary to understand the extensions for blocking resistance are mentioned.

AN.ON [7] (see Figure 3) is based on Mix cascades (or cascades for short). A cascade is a static chain of Mixes [9]. One can understand a Mix and a whole cascade as a black box which forwards messages (or data). This is done in a way so that nobody knows which incoming message belongs to which outgoing message. In this way the communication relations between senders and receivers are hidden. The only way to deanonymize a user of a certain Mix cascade is to control all Mixes of that cascade.

Every user needs a client software called JAP which acts as an interface to the Mix cascades. JAP prepares all communication according to the Mix protocol (splits it into Mix packets; does encryption/decryption etc.) A third component is a distributed database called InfoService which stores information about available Mix cascades and their states



**Figure 3: Architecture of the AN.ON system**

(belonging Mixes, their operators, number of users, traffic situation etc.) Depending on this information a user chooses the Mix cascade he wants to use (typical the one he trusts most).

According to the existing AN.ON system the easiest way to block the communication between a user and AN.ON is blocking all communication with a definite IP address or a definite port-number. A blocker can evaluate the information about cascades provided by the InfoService and extend automatically his blacklist of blocked communication.

Besides, the cascade-based AN.ON system JAP implements a second protocol and can utilize the free routes based anonymizing network called Tor [12]. It is even possible to concatenate both of them. All this can be used to strengthen the anonymity provided to the users. For simplicity, the following explanations consider only the case where a cascade is used as anonymizing service, but it can be easily adopted for the case of using Tor.

### 6.1 Many access points through an add-on network

As the JAP clients are widespread over the Internet community many possible access points to the AN.ON system become available. If a user with restricted internet access (called  $JAP_B$ ) wants to connect to the AN.ON network he contacts a user with free internet access (called  $JAP_R$ ).  $JAP_R$  serves as a router for  $JAP_B$ . He accepts a request from  $JAP_B$  and transparently forwards all communication data between  $JAP_B$  and a chosen Mix cascade.

This approach seems to be similar to the TriangleBoy add-on for the SafeWeb service mentioned earlier. But TriangleBoy is a stand-alone software and a volunteer has no additional benefit from installing it. In contrast to TriangleBoy our approach is to integrate routing functionality into the existing JAP client software. This has the advantage that every JAP user can become a volunteer “out of the box” (without installing any additional software). Each user can decide if and how much bandwidth he provides to blockees. Another advantage is that the backbone of the AN.ON system (Mix cascades and InfoService) can be leaved untouched (only some additional functionality is needed for the InfoService).

Besides  $JAP_R$  and  $JAP_B$  communicate over a lightweight protocol.  $JAP_B$  decides itself which Mix cascade he wants to use. It generates the Mix packets for this cascade and sends them to  $JAP_R$ .  $JAP_R$  transparently forwards them directly

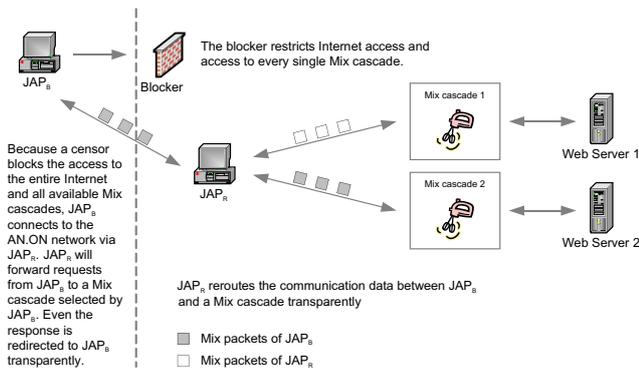


Figure 4: Using the JAP as a forwarder

to the selected cascade. He receives the reply packets and redirects them back to  $JAP_B$ .

From the anonymity point of view  $JAP_R$  acts as a normal Internet router thus the trustworthiness of  $JAP_R$  has no influence on the anonymity of  $JAP_B$  and vice versa.  $JAP_B$  is able to select a Mix cascade he trusts.<sup>7</sup> Furthermore the extension can be easily implemented in the existing AN.ON system.

## 6.2 Using JAP for routing traffic

To meet our needs we have to integrate an additional routing procedure into the existing software. With this a normal JAP client can act as a server, serving requests from JAPs with restricted internet access. However we have to respect additional security constraints so that a volunteering  $JAP_R$  has no drawbacks or runs into trouble. The forwarding should be restricted to Mix cascades only which  $JAP_R$  knows, making it impossible to misuse a  $JAP_R$  for downloading copyright protected material, mounting denial of service (DoS) attacks etc. Second  $JAP_R$  has to have full control over the resources he offers for blocking resistance including bandwidth, amount of traffic, computing power etc.

In order to manage the communication between  $JAP_B$  and  $JAP_R$  a (simple) protocol is needed. The technical details of a typical communication sequence and of the XML based protocol are given in Appendix B. Generally the communication sequence can be divided into three main parts.

1. *Open*  $JAP_B$  establishes a connection to an available  $JAP_R$  and tells him the chosen Mix cascade.  $JAP_R$  answers with the offered resources.
2. *Transmission*  $JAP_R$  transparently forwards data from  $JAP_B$  and back.
3. *Close*  $JAP_B$  sends a message to  $JAP_R$  indicating that he wants to close the connection. Both parties are now able to gracefully shutdown the communication process. Furthermore the connection can be closed by an initiative of  $JAP_R$ , too.

You may have noticed that the explained communication scenario applies to the case where  $JAP_B$  establishes a connection to  $JAP_R$ . However it can be easily adopted for the

<sup>7</sup>In order to reduce the risk, that  $JAP_R$  will offer only compromised cascades,  $JAP_B$  should collect as much information as possible about available cascades immediately after getting access to the “free” Internet. Based on this new information  $JAP_B$  should reconsider which cascade to use.

case where  $JAP_R$  will contact  $JAP_B$  (only the initial step needs to be changed).

Also we are aware of the fact that using plain TCP/IP as transport layer for the protocol between blockee and volunteer may not be the best solution because a blocker could detect this kind of communication by analyzing the information flow. However we want to stress that on the one hand the protocol is independent from a concrete transport layer. Many different solutions are thinkable like SSL tunnel, SMTP, steganography etc. On the other hand (as mentioned in the introduction) we are interested in learning how and how fast a blocker adapts his blocking strategies. For that reason we start with a weak solution which can be strengthened if necessary.

## 6.3 Distributing the necessary information

As analyzed in Section 5, as a consequence of utilizing the “many access points” concept we have to solve a second problem: the distribution of information about available access points. The most open questions and problems are still in this area. Some of them come from the fact that we are limited in resources for our real world experiments. Even if broadcast of information via satellite or short wave may be a good solution we do not have the capabilities to do it. Therefore (and for the reasons mentioned in the introduction) we will start with a weaker solution and try to strengthen this if necessary.

As main source of information we will use the already existing InfoService which is a distributed database. Thus it already owns multiple access points making blocking more difficult. Also new nodes could be added easily (in contrast to establishing new Mix cascades). Every JAP already communicates with that InfoService so implementing the extensions is not that much expensive.

As described above “communicating with the InfoService” does not necessarily mean that a plain TCP/IP connection is used as again the protocol is independent from the underlying transport mechanism. For our first experiments we will also offer a SMTP based solution. From a theoretical point of view this may be only a weak solution but in the real world, blockees write us e-mail telling that the AN.ON servers are blocked.

Every  $JAP_R$  which is willing and able to receive connection requests from a blockee creates a single CAPTCHA with the access information (IP address, port number etc.) and sends this together with a random number to the InfoService. This number is used for “keep alive” messages. As long as  $JAP_R$  provides his service he periodically sends that random number to the InfoService. Otherwise the InfoService will delete the CAPTCHA from its database.

If a  $JAP_R$  is not able to receive incoming connection requests (due to firewall/NAT problems; see below) he periodically checks the InfoService for blockees waiting for help.

Each time a blockee wants to communicate with the InfoService (requesting available  $JAP_R$  or publishing his request for help) he first has to solve a CAPTCHA generated by the InfoService. This ensures that a human not a machine is talking to the InfoService.

Because our system is publicly available for some years we got a lot of e-mails from blockees asking for assistance. We will use this contacts (besides the usual publishing systems like web servers etc.) to publish the necessary initial information about our blocking resistant system. We hope

to achieve the critical mass so that this information will be propagated among the blockees.

## 6.4 Usability of the extended JAP

We want to stress that usability is a big point (maybe the big point) of our system. Anonymity systems need many users because otherwise they could not provide anonymity at all. Blocking resistant systems which are based on the “many access points” concept are depending on many volunteers, too. Thus our system has to be as easily usable as possible and must not discourage potential users ([7], [5], [4]). On the other hand each user has to keep full control of that is going on. This may result in lots of confusing configuration parameters. The upgrade of the existing JAP client has to extend the user interface in a way so that any user can easily manage, activate and use the new functions. This implies that the user needs to understand how the system works. Below we discuss some more usability problems our system may have and propose solutions.

The routing functionality has to be activated manually by each JAP user. We chose this “opt-in” model because we strongly believe that many users would otherwise not be aware of this new functionality and its possible drawbacks. In any case it is better to lose some potential volunteers who not recognize this “opt-in” functionality than having surprised and irritated users wondering what their JAP is doing. Indeed there are some facts a JAP<sub>R</sub> voluntary should be informed about:

- He is limiting his own network resources by providing routing functions to other JAPs.
- He should be aware to check his firewall and general network settings to not compromise the correct routing functionality.
- He should know that his service could be used by a JAP<sub>B</sub> to perform prohibited operations.

Based on this information, every JAP<sub>R</sub> voluntary has to decide if he really wants to provide this service to others.

Besides, also the blockee has to be informed that bypassing the blocking may be risky. The blocker could detect these attempts and prosecute him.

We expect that two network technologies have big influence on the usability and thus on the success of our blocking resistant system. These are firewalls and NAT which are more and more used by everybody. Both will restrict the reachability of JAP<sub>R</sub>. Why should a volunteer spend plenty of time reconfiguring his operating system getting a less secure system at the end? Of course we could try to give assistance to volunteers explaining how they could only open the necessary port but the growing number of different firewalls make this impossible. NAT makes the configuration even more difficult because by default the computers within the private network are not visible to the outside world at all. Of course solutions exist but to burden potential volunteers who get no benefit seems not to be feasible.

One solution of this problem is that JAP<sub>R</sub> will become the initiator of the connection with JAP<sub>B</sub> as described more generally above. Of course now JAP<sub>B</sub> needs to care about his firewall/NAT settings. However we expect that he is willing to do this because he gets a high benefit. In addition we will integrate a “self test”. For self testing a JAP<sub>R</sub> connects to its own server and tries to establish a connection to

a particular Mix cascade. Of course a meaningful self test needs support from a node (server) at the Internet. If there are problems the user will be informed and he can refine the communication parameters.

Before a user activates the routing module, he can specify the communication parameters. He can set the number of concurrent connections he will accept from different JAP<sub>B</sub> at the same time. Besides he can define the up- and downstream bandwidth for communication. Generally a user sets these parameters manually. Considering that there are some JAP users without great technical understanding we will provide presets depending on their Internet connection.

Using the JAP as a router should not depend on communicating anonymously over the AN.ON network. A voluntary user should have the possibility to start a JAP client and only to activate the routing part, without activating an anonymous connection. This possibility is promoted by the chosen design, because every single JAP<sub>B</sub> generates its own Mix packets and routes its traffic transparent to a chosen Mix cascade.

It is absolutely necessary that a JAP<sub>R</sub> has to be informed about the use of the provided forwarding functionality. This is true simply because it is the only benefit a volunteer gets: recognizing that his volunteering jobs is really useful. Ask yourself: wouldn't it be much more fun if you realize that your help is welcome and useful? For this reason we will present as much information as possible emphasizing the “usefulness” of the job a volunteer does. This information includes the number of served blockees (total and at the moment), the number of transferred bytes (per blockee and in total), if possible the region a blockee lives in etc.

## 7. CONCLUSIONS AND FUTURE WORK

We presented a systematics of blocking criteria and developed some general countermeasures. Discussing the pros and cons we come to the conclusion that operating a usable and feasible system which offers blocking resistance even against a strong attacker with huge amount of resources is infeasible. It is more or less simply a race between the blockees and the blocker.

Nevertheless the proposed mechanisms could be used to make the job of the blocker more difficult. Therefore we outlined a concrete design which is based on our existing AN.ON service. Although we know that our system may be weak and will not withstand every blocker we will implement it and make it public. Then the blocker has to make his move. We will learn how and how fast he adopts his strategies. Using this information we will strengthen our system. At the end the blocker will (hopefully) not be willing to spend even more resources on blocking the free access to the Internet.

Nevertheless there are many open questions which need further research. Some of them are pointed out in this paper some of them are not mentioned at all: How does law enforcement influence blocking resistance? Imagine that we do not want an unimpeachable anonymous systems but a system where some information flow can be deanonymized. In this case we have to ensure that this functionality cannot be misused by a blocker to learn which information a blockee consumes. Also we have to ensure that no volunteering forwarder would become liable for the requests initiated by a blockee. He should not even become suspect because otherwise he will stop volunteering. What if the underlying

anonymity service has to be paid? We have to ensure that not the forwarder has to pay for anonymizing the traffic he reroutes on request of a blockee. On the other hand payment can be used as stimulus for volunteering.

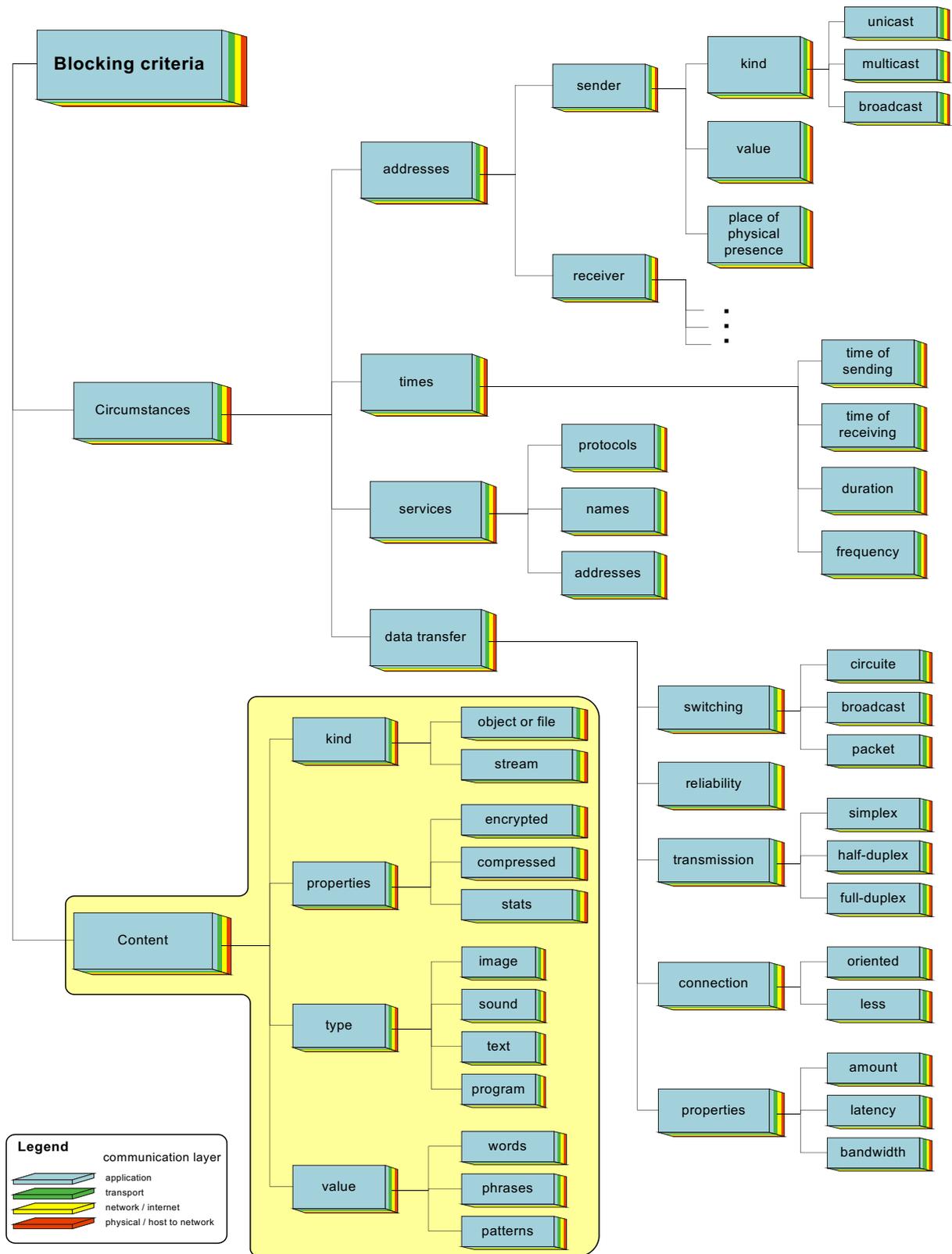
At the very end we want to thank our colleges namely Rainer Böhme, Sebastian Clauß, Hannes Federrath, Thomas Kriegelstein, Andreas Pfizmann and Andreas Westfeld for the interesting discussions (about steganography, sociology, psychology etc.) and for playing the game of being volunteer, blockee or blocker.

## 8. REFERENCES

- [1] About the peekabooby project. the concept and the code.
- [2] Homepage of the peekabooby project. <http://www.peek-a-booby.org/>.
- [3] Homepage of the six/four project. <http://www.brain-pro.de/seiten/six/sixfour.html>.
- [4] A. Acquisti, R. Dingledine, and P. Syverson. On the Economics of Anonymity. In R. N. Wright, editor, *Proceedings of Financial Cryptography (FC '03)*. Springer-Verlag, LNCS 2742, January 2003.
- [5] A. Back, U. Möller, and A. Stiglic. Traffic analysis attacks and trade-offs in anonymity providing systems. In I. S. Moskowitz, editor, *Proceedings of Information Hiding Workshop (IH 2001)*, pages 245–257. Springer-Verlag, LNCS 2137, April 2001.
- [6] K. Bennett and C. Grothoff. GAP – practical anonymous networking. In R. Dingledine, editor, *Proceedings of Privacy Enhancing Technologies workshop (PET 2003)*. Springer-Verlag, LNCS 2760, March 2003.
- [7] O. Berthold, H. Federrath, and S. Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. In H. Federrath, editor, *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, pages 115–129. Springer-Verlag, LNCS 2009, July 2000.
- [8] O. Berthold and H. Langos. Dummy traffic against long term intersection attacks. In R. Dingledine and P. Syverson, editors, *Proceedings of Privacy Enhancing Technologies workshop (PET 2002)*. Springer-Verlag, LNCS 2482, April 2002.
- [9] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 4(2), February 1981.
- [10] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46, 2001.
- [11] R. Dingledine, M. J. Freedman, and D. Molnar. The free haven project: Distributed anonymous storage service. *Lecture Notes in Computer Science*, 2009:67–??, 2001.
- [12] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.
- [13] N. Feamster, M. Balazinska, G. Harfst, H. Balakrishnan, and D. Karger. Infranet: Circumventing web censorship and surveillance. In *Proceedings of the 11th USENIX Security Symposium*, August 2002.
- [14] N. Feamster, M. Balazinska, W. Wang, H. Balakrishnan, and D. Karger. Thwarting web censorship with untrusted messenger discovery. In *Proceedings of the 3rd Workshop on Privacy Enhancing Technologies*, March 2003.
- [15] R. Ferreira, C. Grothoff, and P. Ruth. A transport layer abstraction for peer-to-peer networks, 2003.
- [16] E. Franz, A. Jerichow, S. Möller, A. Pfizmann, and I. Stierand. Computer based steganography: How it works and why therefore any restrictions on cryptography are nonsense, at best. In *Proceedings of Information Hiding, First International Workshop*, volume LNCS 1174. Springer-Verlag, Heidelberg, 1996.
- [17] M. J. Freedman and R. Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, DC, November 2002.
- [18] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Hiding Routing Information. In R. Anderson, editor, *Proceedings of Information Hiding: First International Workshop*, pages 137–150. Springer-Verlag, LNCS 1174, May 1996.
- [19] A. Hintz. Fingerprinting websites using traffic analysis. In *Proceedings of the Workshop on Privacy Enhancing Technologie*, April 2002.
- [20] D. Martin and A. Schulman. Deanonymizing users of the safeweb anonymizing service. In *Proceedings of 11th USENIX Security Symposium*, August 2002.
- [21] R. C. Merkle. Secure communications over insecure channels. *Communications of the ACM*, 21(4), April 1978.
- [22] S. Möller, A. Pfizmann, and I. Stierand. Rechnergestützte steganographie: Wie sie funktioniert und warum folglich jede reglementierung von verschlüsselung unsinnig ist. *Datenschutz und Datensicherung DuD*, 18(6):318–326, 1994.
- [23] M. K. Reiter and A. D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.
- [24] P. Release. Safeweb and voice of america form alliance to free the internet in china. <http://www.safewebinc.com/pr.28.html>.
- [25] M. Rennhard and B. Plattner. Introducing MorphMix: Peer-to-Peer based Anonymous Internet Usage with Collusion Detection. In *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 2002)*, Washington, DC, USA, November 2002.
- [26] W. Stallings. *Data and computer communications*. Macmillan Publishing Company, New York, 4th edition, 1994.
- [27] L. von Ahn, M. Blum, N. Hopper, and J. Langford. Captcha: Using hard ai problems for security. In *Proceedings of Eurocrypt*, 2003.
- [28] M. Waldman and D. Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the 8th ACM Conference on Computer and Communications Security (CCS 2001)*, pages 126–135, November 2001.
- [29] M. Waldman, A. Rubin, and L. Cranor. Publius: A robust, tamper-evident, censorship-resistant and source-anonymous web publishing system. In *Proceedings of the 9th USENIX Security Symposium*, pages 59–72, August 2000.
- [30] D. J. Watts. Six degrees: The science of a connected age. norton, 2003.
- [31] A. Westfeld. F5 – A Steganographic Algorithm: High Capacity Despite Better Steganalysis. In I. S. Moskowitz, editor, *Proceedings of Information Hiding, 4th International Workshop*, volume LNCS 2137, pages 289–302. Springer-Verlag, April 2001.
- [32] J. Zittrain and B. Edelman. Documentation of internet filtering worldwide <http://cyber.law.harvard.edu/filtering/>.

# APPENDIX

## A. CLASSIFICATION OF BLOCKING CRITERIA



## B. TECHNICAL DETAILS OF THE COMMUNICATION PROTOCOL BETWEEN JAP<sub>B</sub> AND JAP<sub>R</sub>

In the first part JAP<sub>B</sub> tries to establish a connection to an available JAP<sub>R</sub> in the following way:

- JAP<sub>B</sub> notices that he is not able to communicate directly with a certain Mix cascade and thus contacts one of the JAP<sub>R</sub> he knows. How a JAP<sub>B</sub> gets an initial list of available forwarders is discussed in the paper.
- JAP<sub>R</sub> receives a connection request from JAP<sub>B</sub>. He decides if he wants to approve this request. If so, he sends a definite response to JAP<sub>B</sub>. If JAP<sub>R</sub> does not accept the connection request JAP<sub>B</sub> has to select another JAP<sub>R</sub>. After a positive response of JAP<sub>R</sub>, JAP<sub>B</sub> is able to select the Mix cascade he wants to use.
- For selecting a certain Mix cascade JAP<sub>B</sub> sends a request to JAP<sub>R</sub>, containing the Mix cascade he wants to use. Now JAP<sub>R</sub> has to decide if the requested cascade is available for use. If JAP<sub>R</sub> makes a negative decision JAP<sub>B</sub> has to select another Mix cascade (or JAP<sub>R</sub>) and retry again. If the Mix cascade is available for use JAP<sub>R</sub> sends a response with determined connection parameters.
- The connection parameters send by JAP<sub>R</sub> inform JAP<sub>B</sub> about the resources (bandwidth, duration, requests per minute etc.) JAP<sub>R</sub> provides to JAP<sub>B</sub>. If JAP<sub>B</sub> disagrees with these restrictions the connection between JAP<sub>B</sub> and JAP<sub>R</sub> will be closed. Now JAP<sub>B</sub> has to find another JAP<sub>R</sub> which provides the requested connection parameters. If JAP<sub>B</sub> agrees with the conditions set by JAP<sub>R</sub>, JAP<sub>R</sub> opens a connection to the Mix cascade selected by JAP<sub>B</sub>. He receives the public connection information of this cascade and forwards them to JAP<sub>B</sub>. With this information JAP<sub>B</sub> is able to generate Mix packets and sends them via JAP<sub>R</sub> to the selected cascade.

After a connection was successfully established the communication process reaches part 2 of the sequence. In this part JAP<sub>R</sub> transparently forwards data from JAP<sub>B</sub> and back. This part of the sequence lasts as long as the connection is open.

Part 3 of the communication sequence concerns closing the connection. Therefore JAP<sub>B</sub> sends a message to JAP<sub>R</sub> indicating that he wants to close the connection. Both parties are now able to gracefully shutdown the communication process. Furthermore the connection can be closed by an initiative of JAP<sub>R</sub>, too. This may happen if the determined connection time for JAP<sub>B</sub> expires or JAP<sub>R</sub> will be shutdown itself. In this case JAP<sub>R</sub> informs JAP<sub>B</sub> that he will close the connection. Now JAP<sub>B</sub> has to find another JAP<sub>R</sub> who is willing to serve as a router for him.

The information exchange between all AN.ON components is based on XML. Figure 6 shows the general structure of XML messages exchanged between JAP<sub>B</sub> and JAP<sub>R</sub> during connection establishment.

The root element is <JAPRouting>. Within this tag we have the possibility to define if the message is a request or a response. Every <Response> resp. <Request> tag has some attributes. The "subject" attribute contains a string representing a certain state of the communication

```
<?xml version = "1.0" ?>      <?xml version = "1.0" ?>
<JAPRouting>                  <JAPRouting>
  <Request subject = "...">    <Response subject = "...">
    msg = "...">              msg = "...">
    ...
  </Request>                  </Response>
</JAPRouting>                </JAPRouting>
```

Figure 5: General form of XML messages used in the protocol between JAP<sub>B</sub> and JAP<sub>R</sub>

process. The "msg" attribute contains a concrete request or response regarding to the reference. Additional tags within the <Response> or <Request> tag may transport additional information.

The whole protocol is as follows:

1. JAP<sub>B</sub> opens a TCP/IP connection to JAP<sub>R</sub>.
2. If JAP<sub>R</sub> is not willing to serve JAP<sub>B</sub> he closes the connection immediately. Otherwise he sends a response containing subject="connection" and msg="accepted". This message tells a JAP<sub>B</sub> that the requested JAP<sub>R</sub> will act as a forwarder for JAP<sub>B</sub>.
3. JAP<sub>B</sub> informs JAP<sub>R</sub> which cascade he wants to use sending the XML structure below.

```
<?xml version = "1.0" ?>
<JAPRouting>
  <Request subject = "cascade" msg = "select" >
    <MixCascade id = "cNewYorkBerlinDresden" />
  </Request>
</JAPRouting>
```

Within the request tag information about the selected Mix cascade is transmitted. Every Mix cascade has a unique identifier ( "id" attribute). Based on this id information a JAP<sub>R</sub> can request additional information from the InfoService about this cascade and decide if JAP<sub>B</sub> can use the requested Mix cascade. The special id="\*" is used to signal JAP<sub>R</sub> that he can chose a cascade on his own. This may be necessary if JAP<sub>B</sub> has no ideas about available cascades. After establishing a connection to the AN.ON network JAP<sub>B</sub> can use this to request information about available Mix cascades from the InfoService. He than closes the current connection and establishes a new one with a Mix cascade of his choice.

4. If JAP<sub>R</sub> rejects this cascade he sends a response with subject="cascade" and msg="notallowed". In this case JAP<sub>B</sub> has to select another Mix cascade. If JAP<sub>R</sub> allows communication to the chosen Mix cascade he responds with a confirmation containing a contract of resources he is willing to offer to JAP<sub>B</sub>. The <Contract> element specifies the up- and downstream bandwidth (in Mix packets per minute) and the duration for which JAP<sub>B</sub> can use JAP<sub>R</sub>. Additional limitations are thinkable and could be specified later.

```
<?xml version = "1.0" ?>
<JAPRouting>
  <Response subject = "contract" msg = "confirm">
    <Contract>
      <Downstream>...</Downstream>
      <Upstream>...</Upstream>
      <Duration>...</Duration>
      ...
    </Contract>
  </Response>
</JAPRouting>
```

5. JAP<sub>B</sub> has to confirm this contract which assigns the network resources JAP<sub>R</sub> is willing to provide. If JAP<sub>B</sub> agrees everything is fine and the transparent communication channel to a Mix cascade can be used. Otherwise JAP<sub>B</sub> closes the connection and has to find another JAP<sub>R</sub> willing to serve his requests.

```
<?xml version = "1.0" ?>
<JAPRouting>
  <Response subject = "contract" msg = "accepted" />
</JAPRouting>
```

To gracefully shutdown a connection JAP<sub>B</sub> or JAP<sub>R</sub> sends a request message with **subject="connection"** and **msg="close"**.

It is obvious that the suggested XML protocol is very easy to extend. More functionality can be added if needed. Through the combination of using request and response tags with different attributes as well as special tags for the delivery of information our protocol becomes very flexible.