A Round-Based Cover Traffic Algorithm for Anonymity Systems

Marta Rybczyńska

Abstract. Anonymity is becoming more of an issue with the growing importance of networking. Examples include protecting privacy of Internet users or electronic voting. Several network anonymity systems have been deployed, the most popular of them is probably Tor. However, those systems do not protect against attackers who observe or modify the traffic to match sources with destinations. The protection method against such attacks by adding additional traffic is not usually implemented because of the high cost. In this work we propose a new cover traffic generation algorithm for flow-based anonymity systems and compare it with other algorithms from the literature. Our algorithm is based on four ideas: fixed time rounds, flow classification with different protection methods for different classes, protection depending on the potential cost and finally, use of history. We evaluate our algorithm both in theory and in practice. Our analysis show that that our solution provides sufficient protection while reducing overhead traffic compared to the algorithms known from the literature.

1 Introduction

There are many definitions of anonymity. A short definition can be found in the anonymity terminology collected by Pfitzmann and Köhntopp [18], where anonymity is defined as the state of being not identifiable within a set of possible subjects (also called the anonymity set). Anonymity system that provides better anonymity to its' users is typically built as an overlay network over standard Internet architecture.

Marta Rybczyńska
Institute of Telecommunications,
Warsaw University of Technology,
Warsaw, Poland
e-mail: `marta@rybczynska.net`

Anonymity systems use various techniques to protect users' privacy, such as cryptographic transformation, introducing artificial delays, and reordering or adding additional (or cover) messages. Impressive effort has been put into improving the designs, but deployed systems are still vulnerable to certain attacks: protective mechanisms are either incomplete or require too many resources to be practical. A recent overview of existing designs and deployed systems may be found in a survey by Danezis and Diaz [6].

So called 'timing attacks' (or 'timing analysis') are probably the most important class of such attacks. The attacker observes messages in the network and searches for timing correlations to determine communication paths. Such attacks can be either active or passive.

In this work, we propose a new algorithm for cover traffic generation that provides decent protection against timing analysis while limiting the protection cost compared to existing solutions. Our algorithm also offers possibilities of even better protection. We compare our proposal with other algorithms from the literature. Our results are based on a theoretical analysis and simulations of a set of nodes with different traffic distributions, including a set of effective attacks proposed by other authors.

We begin by providing background on cover traffic algorithms in anonymity systems, known protections, attacks and the methods of calculating anonymity (Section 2). We model our anonymity system and attackers in Section 3. Section 4 presents our algorithm. We start from basic concepts, then move to the core algorithm and possible extensions. In Sections 5 and 6, we evaluate our algorithm. First we perform a theoretical evaluation using the degree of anonymity metric. Later we present and discuss simulation results in terms of protection and cost. We conclude in Section 7.

2 Previous Work

Cover traffic is additional traffic added into an anonymity system to make it harder for an attacker to trace flows in the system by making them more similar to each other. There is significant literature on the subject, but the deployed and popular anonymity systems like Tor do not use this protection method because of the high cost [9]. We review cover traffic in the first part of this section.

The second part is devoted to the methods of evaluating the protection level given by an anonymity system. We provide details on the most popular metric, the degree of anonymity.

2.1 Protecting Anonymity Systems Using Cover Traffic

Anonymity systems have used cover traffic (also called 'dummy messages') as a protection method almost from the beginning. Notable examples include ISDN-Mixes [19], which use ISDN channels.

Cover traffic also appears in the earliest Internet-based designs, like PipeNet, where all clients transmit at the same rate [5]. The details were not specified, however. In the Web MIXes, each client always sends the same number of messages per time slice, inserting dummies if it has no data to send [3]. Tarzan uses a different approach, where cover traffic is generated in small sets of nodes [10].

Fu, Graham, Bettati and Zhao [11] and Levine, Reiter, Wang and Wright [14] show that even constant-rate cover traffic is vulnerable to traffic analysis under certain assumptions. Fu et al. perform experiments and find such vulnerability in practice. They explain it at the low level by subtle timer interrupt delays caused by packets with user data. They do not perform further experiments to check if other operating systems are vulnerable and how (and if) this problem appears on a system with high network load.

Levine et al. [14] assume that cover messages are added only by end nodes. Network losses or the attacker may then introduce holes in the traffic and the holes may then be used to correlate flows. Levine et al. introduce an algorithm called 'defensive dropping'. Intermediate nodes are instructed to drop certain messages and this changes the flow characteristic. They also propose a method to cross-correlate between output flows and input flows. The sequences to be compared are calculated by counting the number of received messages during fixed time intervals. We use the same method in this work and explain it in more detail in Section 6.1.5.

In a Freedom system security analysis, Back, Goldberg and Shostack [2] state that cover traffic was not included because of unresolved issues. Dingledine, Mathewson and Syverson express a similar opinion in the paper on Tor design, stating that the benefits are unclear while the costs are high, and await a design that will defend against a realistic adversary [9].

Zhu, Fu, Graham, Bettati and Zhao [27] study flow correlation attacks in traditional mix systems. They analyse different mix types using mutual information and frequency analysis to correlate flows, and show that existing designs are vulnerable. They propose an algorithm that inserts cover messages into all output flows of a node with exactly the same timing, but buffers received messages for a short time to limit the number of additional messages slightly.

Shmatikov and Wang [22] present an adaptive padding algorithm, where additional messages are inserted into statistically unlikely gaps in the flow. They use a traffic distribution from real-world traffic. As the authors note, the algorithm may provide less protection if the distribution is different from the assumed one. The authors mention the possibility of dynamically modifying the distribution, but they do not discuss this further.

Some studies have examined the methods the adversary may use. An interesting example is the work of Yu, Fu, Graham, Xuan and Zhao [26], who propose a signal watermarking technique based on Direct Sequence Spread Spectrum (DSSS) that uses a Pseudo-Noise (PN) code. The attacker adds a secret spread spectrum signal by slightly changing the sender's traffic rate and recovers the signal after the flow is transmitted to the receiver.

2.2 Measuring Anonymity

The anonymity definition by Pfitzmann and Köhntopp we show in Section 1 defines anonymity using anonymity set. The metric based on anonymity set was used in previous studies, but it requires a method to decide if a certain node should be included in the set or not. For instance, the works of Kesdogan, Agrawal and Penz [12] and Kesdogan, Egner and Büschkes [13] define anonymity set in such a way that they include a node into the anonymity set if the probability of the desired role of the node (sender or receiver) is higher than zero. Their definition and the definition of anonymity set assumes that the decision of including a certain node in the anonymity set is always binary: yes or no. It may be, however, that different senders and receivers may send or receive messages with different probabilities. The anonymity set metric does not take this into account.

A popular anonymity metric that solves this problem has been independently introduced by Serjantov and Danezis [20] and Díaz, Seys, Claessens and Preenel [8]. It is calculated under an assumption of a specific attack. If, after an attack, an attacker can assign probabilities p_i to the senders and the number of senders is N, then we can calculate the entropy:

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i). \tag{1}$$

It can be interpreted as the number of bits the attacker must learn to determine the identities of the senders or the receivers.

Díaz et al. introduced a metric using $H(X)$ and the maximum possible entropy of a set of N denoted as $H_N = log_2(N)$[1] introduced for normalisation. The metric is called the degree of anonymity and is defined as follows:

$$d = \frac{H(X)}{H_N}. \tag{2}$$

They also proposed an extension to their metric when the analysis must cover multiple cases and the probabilities differ in each of those cases. Their first proposal was to calculate a weighted sum of degrees of anonymity:

$$d = \sum_{j=1}^{K} p_j d_j, \tag{3}$$

where K is the number of cases, p_j the probability of certain circumstances and d_j is the degree of anonymity under these circumstances. The alternative method is to compare the calculated degrees with a certain assumed threshold and raise an alarm if the anonymity is below the threshold.

[1] The original work of Díaz et al. uses H_M for maximum entropy. We prefer to use H_N to denote the maximum entropy for N nodes.

This method is used in practise to evaluate anonymity system designs. For instance, Serjantov and Newman use this method to analyse the anonymity of timed pool mixes [21]. The same method is later used by Steinbrecher and Köpsell [23] to measure unlinkability.

The recent work of O'Connor [17] formally shows, using the entropy-based metric, how a message-based system using traditional mixes degrades over time: geometrically after certain threshold. The analysis is performed only for message-based systems.

The degree of anonymity was later analysed by Tóth, Hornák and Vajda [24] who highlighted situations where the calculated degree of anonymity is the same, but the protection given to a particular party varies greatly. They also provide examples of such cases. The simplest example has two anonymity systems: in the first one probability is the same for all participants. In the second system, however, the probability for the actual sender is 0.5 and for the others it is distributed equally. It turns out that we can find the number of nodes in both anonymity systems that the entropy is the same, for instance 20 nodes in the first anonymity systems and 101 nodes in the second one. Obviously the protection provided by those systems, as seen by the users, is not the same.

In this work we use the degree of anonymity metric to evaluate our algorithm. The problems highlighted by Tóth et al. are valid and should be taken into account when analysing the results, but at this time there is no other widely accepted anonymity metric for anonymity systems. Note also, that we use a weighted degree of anonymity that takes into account all possible cases.

3 Assumptions and Models

We assume that the anonymity system consists of a number of nodes that exchange fixed-length encrypted messages. As each node re-encrypts (and possibly modifies) the messages, an attacker cannot distinguish between messages based on the content. That leaves the inter-arrival times of the messages as a source of attacks.

The anonymity system transmits data in organised flows. Intermediate nodes can distinguish the flows in order to route them correctly. We do not cover the algorithm for route selection and establishment in this work: we assume that this phase has already been completed, and all flows have been successfully established. All messages have the same size.

Our attacker is very powerful. Not only can it monitor all the links between the nodes, but it also has the power to modify traffic by adding artificial delays to chosen messages. In our analysis, the adversary tries to match source and destination nodes by correlating the number of sent and received messages on selected links. All the algorithms used in the system and the structure of the network are known to the attacker, who can also distinguish between flows, even if they use the same link.

4 A Class-Based Cover Traffic Algorithm

4.1 Core Ideas

We design our algorithm to be used in an anonymity system where available bandwidth is limited and is a valuable resource. It also means that it may not be possible to fully protect every transmission in that anonymity system. There should be, however, a method to notify users of the current estimated protection level.

The algorithm presented in this work is based on four ideas: fixed time rounds for calculating the amount of cover traffic, different flow classification and protection methods for different classes, history information, and finally, the observation that certain flows are harder to protect than others.

4.1.1 Rounds

The basic unit of time is one round. During a round, a node receives and buffers messages, while sending messages from the previous round. At the end of each round, nodes add cover traffic messages to each flow. Then, scheduling takes place and a new round begins. The length of a round is a system-wide constant.

The existence of rounds introduces delays. The delay, if used with an admission control algorithm, is constant and may be changed at network configuration time.

The main reason for introducing rounds is the possibility of calculating the required amount of cover traffic from the real traffic statistics, and scheduling all messages equally during the next round. There are other reasons as well: for example, a slight change in inter-packet times during a round.

4.1.2 Flow Classification and Class-Based Protection

Flows are classified by their bandwidth and changes in bandwidth usage during subsequent rounds. When there are different traffic classes, each class may be protected in a different way. For instance, low-bandwidth classes may be protected more than high-bandwidth ones. The rationale for such behaviour is simple: less cover traffic is required to change all flows in a low-bandwidth class to fit the same (also low-bandwidth) pattern.

With more than one traffic result pattern, as in the traditional solutions, changing the flow to fit the closest pattern from a set of available ones should require fewer resources.

Introducing flow classification and class-based protection parameters has one drawback, however. Each class should contain roughly the same number of flows. Otherwise, the attacker may use the fact that the flow is one of only a few in its class.

4.1.3 Use of History

Our algorithm uses flow history rather than only using the flow class from the current round. This helps smooth rapid changes (like holes in the traffic) caused by, for instance, congestion. It also limits space for the attacker, as changes in flow parameters will be smoothed within the history length.

4.1.4 Different Levels of Protection for Different Flows

If there are major differences between the flows (for example, in the amount of bandwidth used), it may be more expensive to protect some flows than others.

Thus, we choose to differentiate protection. We offer better protection to lower-bandwidth flows, as this requires fewer additional messages. Such behaviour requires notifying users of the situation and, for instance, negotiating lower bandwidth if they require higher-grade anonymity.

4.2 The Basic Algorithm

Nodes process received messages during so-called rounds. Packets received during a round are buffered. When the round ends, for each flow we calculate the maximum number of packets received during a fixed number of previous rounds (the history is taken into account here), including the current round. If there are buffered messages from the previous rounds, we add their number to the maximum. Then we compare the value with the maximum bandwidth of the flow. If the maximum bandwidth is greater than or equal to the number of messages, then we classify the flow using the second value. On the other hand, if the number of messages received is greater than the maximum bandwidth, we classify the flow as its maximum bandwidth and put the last messages above the maximum threshold into the buffer. If the buffer fills up during that process, then we drop the remaining messages.

Let i be the current round and j be the current flow. $in[j][i]$ denotes the number of input messages for flow j during round i and $out[j][i]$ denotes the number of output messages for flow j during the round i. $B_{max}[j]$ is the maximum bandwidth for the flow j, and H_{len} is the history length. $M[j]$ is a temporary variable that stores the maximum number of messages for flow j received during the last H_{len} rounds. $class(N, j)$ is an algorithm that finds the class of the flow from the number of messages N for the flow j, and returns the total number of messages that should be sent during the next round.

The table $in[j][-H_{len}]..in[j][0]$ is initialised with $B_{max}[j]$ to hide the characteristics at the beginning of the flow and to protect very short flows.

We can then decide how many cover messages to insert using the algorithm 1 (with the details of handling messages above the limit omitted for clarity's sake).

An example of the $class()$ function is presented later in Section 6.

Algorithm 1. onRoundEnd(i,j), calculate the amount of cover traffic for the given round i and the flow j

```
Require: $i \geq 0, j \geq 0$
  $M[j] = max(in[j][i-H_{len}]..in[j][i-1])$
  if $M[j] \leq B_{max}[j]$ then
    $out[j][i] = class(M[j], j)$
  else
    $out[j][i] = class(B_{max}[j], j)$
  end if
```

4.3 Extensions

Numerous extensions to the basic algorithm are possible. For instance, we may want to increase the length of the flow of more than H_{len} rounds beyond what the basic algorithm does. Such an extension would let us hide the flow's end and possible short-term gaps. Additionally, the increase of length should not be easily predictable. For such cases, we propose an extension that alters the flow based on the rate at which the flow uses allowed and allocated bandwidth. Flows using less bandwidth have their flow length increased more.

In addition to the variables used in the basic version of the algorithm, $fill\_class(f, j)$ is a function that uses the flow parameters and the ratio of used bandwidth f for the flow j to return the base of the increase in flow length. The table $fw[j]$ stores the base increase for the flow j and $flow\_wait[j]$ stores the number of rounds that remain for the flow j. The calculation of the increase additionally uses the default increase D, which is a system constant. $rand(X)$ returns a random integer value in the range of $[0, X)$.

Algorithm 2. onRoundEnd(i,j), calculate the amount of cover traffic for the given round i and the flow j (extended version of algorithm 1)

```
Require: $i \geq 0, j \geq 0$
  $M[j] = max(in[j][i-H_{len}]..in[j][i-1])$
  $S = 0$
  for $k = (i - H_{len})$ to $(i-1)$ do
    $S = S + in[j][k]$
  end for
  $f = (S * 100)/(H_{len} * M[j])$
  if $M[j] \leq B_{max}[j]$ then
    $out[j][i] = class(M[j])$
  else
    $out[j][i] = class(B_{max}[j])$
    $fw[j] = fill\_class(f, j)$
  end if
```

Algorithm 3. onCoverGeneration(i,j), add additional cover traffic to hide the temporary lack of traffic for the given round i and the flow j

```
Require: $i \geq 0, j \geq 0$
  $c = fw[j] - out[j][i]$
  if $c \leq 0$ then
    return
  end if
  if $out[j][i] = 0$ then
    if $flow\_wait[j] > 0$ then
      $flow\_wait[j] = flow\_wait[j] - 1$
    else
      $flow\_wait[j] = 0$
    end if
  else
    $flow\_wait[j] = rand(fw[j]) + D/2 + 1$
  end if
  $out[j][i] = out[j][i] + c$
```

The pseudo-code shown in algorithms 2 and 3 describes our algorithm (with the details of handling messages above the limit omitted for clarity's sake, as in the previous version).

An example of the $fill\_class()$ function is presented later in Section 6.

4.4 Discussion

The idea of rounds is similar to the mix time-out from timed mixes [7], but the details and purpose are different. During a round, messages between two nodes are transmitted in the same order in which they were received, while mixes may change that order. The purpose of introducing rounds is to add the option of adding cover traffic messages at equal intervals between the data messages. This would not be possible without knowing the number of messages received during the round. We introduce rounds as a feature that enables protection, not as a protection mechanism itself.

Uniform distribution of packets during rounds and scheduling is an essential part of the proposed solution. It destroys the low-level packet inter-arrival time distribution, but should not introduce additional artifacts. It is also important to take into account that observers will not be synchronised with the round clock, so they may get different counts of packets during rounds. When doing tests, we had an implementation that did not use the uniform distribution and simply scheduled cover messages at the beginning of the round. Especially for longer flows, it generated artifacts that were sufficient for the attacker to find a good correlation.

The mechanism of history and flow classification based on the maximum value in the history window limits the number of events of bandwidth change, as the attacker may use such events to correlate flows. The same mechanism allows fast recovery of high use of bandwidth.

It should be noted that the algorithm is designed to limit the number of cover traffic messages in common situations. In the worst case, if the attacker sends B_{max} during one round every H_{len} messages, the algorithm will work just as the full cover traffic. This situation may be detected however, using only limited additional resources, as the history is available for each node.

The algorithm does not hide the connection start. The cover traffic mechanism does not seem to be the right place for that protection as one of our requirements is low delay. Another mechanism should be used to prevent the attacker from using the timings of flow starts. It may be performed for instance by predicting user activity and starting a new connection earlier.

In the above pseudo-code, we assume that the number of messages received during a round does not exceed the maximum bandwidth. We made this simplification in order to keep the description and pseudo-code clear. In real implementation, it would be easy to add buffering of additional messages. However, the implementation details will depend on the transport protocol used, as it determines whether messages may (or should be) dropped and whether flow control algorithms should be used (and if so, which ones).

5 Theoretical Evaluation

In this section we present a theoretical evaluation of our algorithm using the degree of anonymity metric explained in Section 2.2. Both simple and extended versions of the cover traffic algorithm are modelled here, but the later one with the assumption of the same extension given to all traffic classes.

5.1 Modelling the Anonymity System

We assume an anonymity system with N senders and N receivers. The input and output links of the anonymity system are observed by the attacker.

The possible observed values come from the alphabet $\mathscr{A}$. The alphabet may be interpreted differently depending on the attacker's capabilities. For instance, the attacker may only have information if any messages occurred in the given time interval. Then $\mathscr{A} = \{0,1\}$. On the other hand, the alphabet may be directly interpreted as the number of messages observed, or as an interval identifier.

Observations from subsequent time intervals form sequences. S_i denotes the sequence observed for the node i, $S_i = \{s_1, s_2, ..., s_L\}, s_i \in \mathscr{A}$. The observed sequences have the length L. Let us denote $\mathscr{S}$ as the set of all possible sequences of length L.

Each of the senders I_i outputs one sequence S_i. Those sequences are then processed by the anonymity system and result in output sequences R_j received by the receivers O_i.

5.1.1 Assigning Probabilities

We denote $P(S_i)$ as the probability that the input sequence is S_i. $P(R_j|S_i)$ denotes the probability of the output (received) sequences R_j if the sent sequence was S_i.

Our formulas include the probability of each sequence $P(S_i)$, so it can be used with any traffic distribution. However, for comparisons between different cover traffic algorithms, we choose to use a traffic distribution that produces every sequence with equal probability. That allows us to remove the effects caused by the assumptions about the traffic distribution and analyse all cases, even those that would have a very low probability in practice, but may be used by the attacker.

Given the traffic distribution that produces our inputs and the model of the anonymity system, for each observable input sequence we can calculate all possible outputs with their probabilities:

$$M_i = \{S_i, P(S_i), \{\langle R_0, P(R_0|S_i)\rangle, \langle R_1, P(R_1|S_i)\rangle, ..., \langle R_N, P(R_N|S_i)\rangle\}\}. \tag{4}$$

With the tuples calculated for each input sequence S_i we can calculate the receiver and sender anonymity, the number different output sequences and the number of receivers with the same sequence.

Iterating all possible input sequences S_i we can calculate the receiver anonymity from the formulas (1) and (2).

For the analysis, the number of alphabet symbols is reduced to 2 (except for the analysis of the dependency on the alphabet size). The following parameters are taken into account: sequence length L, history length H_{len}, the alphabet size and input traffic distribution. For the extended version of the algorithm the list also includes the maximum extension length.

As the calculations are computationally intensive, they have been performed for the ranges of parameters where the results are expected in acceptable time.

5.1.2 Finding Output Classes

We can get the number of output classes directly by searching for all observable R_j. The probability that a node belongs to a specific class can be calculated as follows:

$$P(R_j) = \sum_{i=1}^{N} P(R_j|S_i) * P(S_i). \tag{5}$$

We can also calculate the number of elements in each class R_i given the number of inputs and their sequences.

5.1.3 Calculating the Degree of Anonymity

Let us assume, that the attacker knows the sender and wants to uncover the receiver. The attacker knows the input traffic distribution and all the anonymity system algorithms, so one can assign probabilities $P(R_j|S^*)$ of the anonymity system changing the input sequence S^* to all possible output sequences R_j.

Output nodes can then be divided into groups based on their R_j. Each of the receivers in one group may be the addressee of the sequence the attacker is tracking, and thus the attacker will be unable to distinguish between the nodes in a group without further information. On the other hand, we are able to calculate the probability of each output sequence R_j using the equation (5). It means that we can also calculate the expected size of each R_j group. The expected size of the anonymity set of R_j is then:

$$E_{R_j} = N * P(R_j) = N * \sum_{k=1}^{N} P(R_j|S_k) * P(S_k). \tag{6}$$

Iterating all possible input sequences S_i we can calculate the receiver anonymity from the formulas (1) and (2). The only problem is the number of nodes in each group, as the expected values of E_{R_j} may not be integers. In this work we round the number of nodes in the set $F_{R_j} = round(E_{R_j})$ and assume that the situation we consider is an equivalent to F_{R_j} nodes, each with probability $P(R_j|S_i)/F_{R_j}$ of being the receiver, for $F_{R_j} > 0$. Alternatively, we can calculate the anonymity for the set by either rounding up or down. If E_{R_j} is greater than 0, but less than 1, we assume $F_{R_j} = 1$, as we do not want to ignore the classes of low probability:

$$F_{R_j} = \begin{cases} round(E_{R_j}) & \text{for } E_{R_j} \geq 1. \\ 1 & \text{for } 0 < E_{R_j} < 1 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

However, there is also a need to consider the effect of rounding the size set, as the total number of considered outputs may be higher than N. Rounding changes the input traffic distribution and the number of considered nodes. The increased number of nodes is denoted as M. It is the sum of all expected numbers of nodes over all possible output classes:

$$M = \sum_{i=1}^{N} F_{R_i}. \tag{8}$$

The degree of anonymity d_r may then be calculated as:

$$\begin{aligned} H_i(X) &= -\sum_{j=1}^{N} \sum_{k=1}^{F_{R_j}} \frac{P(R_j|S_i)}{F_{R_j}} * log_2 \frac{P(R_j|S_i)}{F_{R_j}} \\ &= -\sum_{j=1}^{N} F_{R_j} \frac{P(R_j|S_i)}{F_{R_j}} * log_2 \frac{P(R_j|S_i)}{F_{R_j}} \\ &= -\sum_{j=1}^{N} P(R_j|S_i) log_2 \frac{P(R_j|S_i)}{F_{R_j}}, \end{aligned} \tag{9}$$

$$d_r = -\sum_{i=1}^{N} P(R_j) \frac{H_i(X)}{log_2 M}. \tag{10}$$

5.2 *Basic Version*

In the simple version, the algorithm uses only the values from its history and the output is equal to the maximum value in the history window. Therefore, the protection depends on the history length H_{len} and the alphabet size. Generally, we expect that the higher H_{len} is, the higher the protection will be. When it comes to the alphabet, the less symbols, the less different possibilities for the attacker, so the protection should be higher.

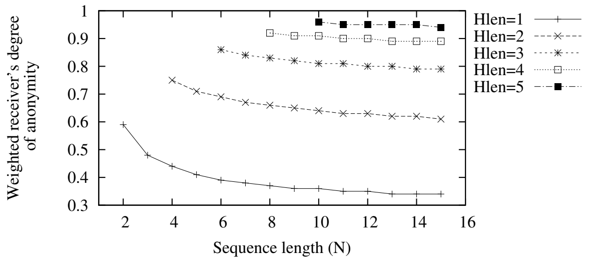

Fig. 1 Receiver's degree of anonymity as a function of sequence length for the basic version of our cover traffic algorithm for different history window sizes H_{len}

Figure 1 shows the degree of anonymity as a function of sequence length for history length from 1 to 5. The values for sequence length less than twice the H_{len} were omitted, as it is heavily dependant on initialisation. It may be seen that increased H_{len} does indeed increase protection and even for relatively short $H_{len} = 2$ the degree of anonymity raises to acceptable 0.7 for long sequences. Additional increases of the value, however, have less and less impact with the increase of H_{len}. This effect

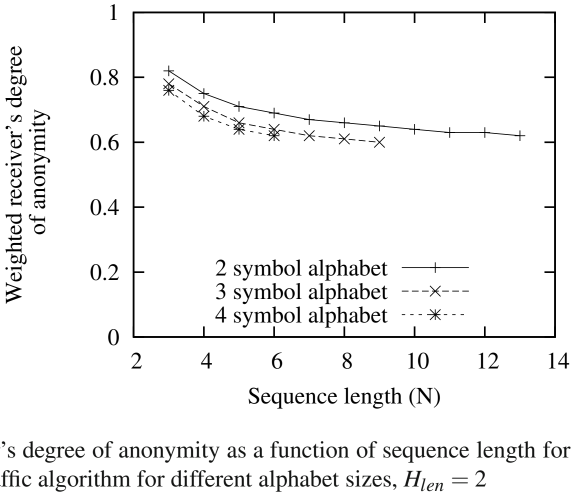

Fig. 2 Receiver's degree of anonymity as a function of sequence length for the basic version of our cover traffic algorithm for different alphabet sizes, $H_{len} = 2$

may be explained by the fact that the number of 'gaps' to be filled becomes smaller and the shorter 'gaps' in the traffic have been already filled at lower H_{len}.

The more symbols in the alphabet, the more possible sequences and also the less protection. The degree of anonymity for 2, 3 and 4-symbol alphabet is shown in Fig. 2. The differences are visible, but small. For the same ratio, the higher number of symbols means much higher number of nodes in rare groups. However, it should be noted, that the number of possible symbols is different as it depends on the alphabet size, so it is 2^i for 2-element alphabet, 3^i for 3-element alphabet and so on.

5.3 Extended Version

The extended version of the algorithm is very similar to the basic one, but includes one modification. The flow may be extended further after H_{len}, for the maximum of D rounds. The length of the extension is an integer value in the range $[1,D)$ with equal probabilities. We denote the situation with no extension as $D = 0$.

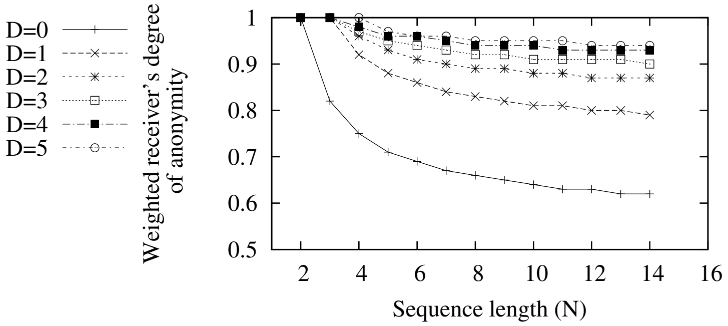

Fig. 3 Receiver's degree of anonymity for different values of maximum extension D, extended version of the algorithm, $H_{len} = 2$

Figure 3 shows the receiver's degree of anonymity for different values of the extension D. The increase of D changes the degree of anonymity in a similar way as H_{len} (compare Fig. 1). The difference is that the increase of H_{len} and D change the number of additional messages to be sent (so the cover traffic cost) in slightly different ways.

The conclusion is that the extended version may increase the protection for a common case, but it introduces artifacts that may be used by the attacker.

5.4 The Influence of Input Traffic Distribution

The input traffic distribution is a parameter of our model, but it is an important factor at the implementation phase. That is why we compare the results for the uniform distribution with a different distribution. The distribution chosen models an ON-OFF source, which may be understood as if there were two main states: sending (ON, 1) or not sending (OFF, 0). This model is relatively simple and widely used in the literature [1, 4] and available in network simulators [15]. The length of its stay in each of the states comes from the exponential distribution, each with independent λ parameter. Figure 4 shows receiver's degree of anonymity for different input

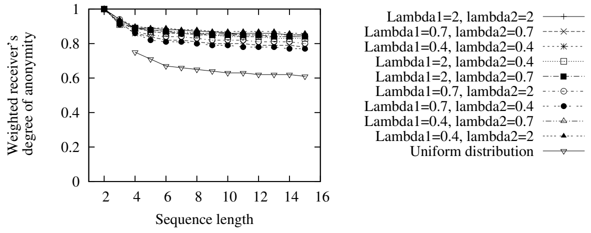

Fig. 4 Receiver's degree of anonymity as a function of sequence length for the basic version of our cover traffic algorithm for different input traffic distributions, $H_{len} = 2$. Lambda1 is used for in the state of non-sending and lambda2 is used in the state of sending.

traffic distribution: uniform like in the previous examples and different versions of the ON-OFF distribution for three values of each of the two lambda parameters of the exponential distributions. It suggests that the algorithm protects ON-OFF traffic better than traffic of uniform distribution. This is related to the assumptions the algorithm was designed under and the mechanism of extending flows (that the flows are visible n the traffic). The differences between the different lambda parameters are small.

6 Practical Evaluation

We implemented the proposed algorithm and compared it with other algorithms from the literature in terms of increased protection and cost (the amount of additional traffic).

6.1 Implementation and Test Configuration

For evaluation, we have implemented and tested our cover traffic algorithms using the ns-2 simulator [16]. We have compared our algorithm with a situation without any cover traffic, with the algorithm of Zhu et al. and with the algorithm of Shmatikov and Wang. It should be noted that we had to modify the two last algorithms to work properly in our test configuration. We present the modifications later in this section.

6.1.1 Simulated Network Configuration

The network configuration consisted of 320 source nodes, 320 destination nodes and a single intermediate node. Each of the source nodes transmitted a single flow to one of the destination nodes. There were the following types of traffic:

- 64 constant rate flows at rates from 32 to 256 kbit/sec (class 1)
- 64 constant rate flows with random intervals between the packets, at rates from 32 to 256 kbit/sec (class 2)
- 64 flows using a Pareto distribution of delay between the packets, at rates from 32 to 256 kbit/sec (class 3)
- 64 flows using exponential distribution of delay between the packets, at rates from 32 to 256 kbit/sec (class 4)
- 16 flows with bursts, burst length and interval of 0.1, 0.2, 0.5 or 1.0 second at 256 kbit/sec (class 5)
- 16 flows with single burst of length of 10, 20, 30 or 40 per cent of the simulation time at 256 kbit/sec(class 6)
- 32 PN (pseudo-noise) sequences with bit length of 0.2, 0.5, 1.0 and 2.0 seconds like proposed in [26] at 256 kbit/sec (class 7)

For each of the five algorithms, the simulation times ranged from 5 to 300 seconds (5, 10, 20, 30, 40, 50, 75, 100, 150, 200, 250 and 300 seconds) and each simulation was repeated 11 times. All packets had data fields of 512 bytes.

Each flow was bi-directional. One of the directions transmitted data, while the other remained silent. If the cover traffic algorithm transmits data on all open flows (which is true for our algorithm and for the algorithm of Zhu et al.), cover traffic was also generated on the silent ones and included in the further cost calculations. The silent flows were not, however, directly taken into account when calculating protection ratio.

Classes 1 to 4 represent different distributions of traffic, while classes 5 to 7 show different methods of possible attacks, where the attacker modifies the traffic to form a known pattern. We use different distributions of traffic, because there is not enough data on the flow characteristic of deployed anonymity systems. Additionally, we assume that different protection of different traffic classes and possible traffic shaping (which we suggest) may change user behaviour patterns, so also the traffic distribution. Flows that differ from the others may be also introduced by the attacker.

6.1.2 Our Algorithm

While testing our algorithm, we used two sets of parameters, each with a round time equal to 0.1 sec.

The first one, 'Classes, $D = 5$' used used four classes for the $class()$ function with $H_{len} = 16$ (in rounds, that equals 1.6 seconds), $B_{max}[i] = 7$ for each i (7 packets/sec, approx 280 kbit/sec) and four classes for the $fill\_class()$ function with $D = 5$ rounds (equalling 0.5 seconds).

The $class()$ and $fill\_class()$ functions used in the simulations may be presented with the pseudo-code shown in algorithms 4 and 5, respectively.

Algorithm 4. class(N,j), return the amount of cover traffic for the flow j after classifying the number of messages N

```
if $N/B_{max}[j] > 0.75$ then
    return $B_{max}[j]$
end if
if $N/B_{max}[j] > 0.5$ then
    return $3 * B_{max}[j]/4$
end if
if $N/B_{max}[j] > 0.25$ then
    return $B_{max}[j]/2$
end if
return $B_{max}[j]/4$
```

Algorithm 5. fill_class(f, j), return the maximum extension of the flow f for round j

```
if $f > 0.75$ then
    return $D * 2 + 1$
end if
if $f > 0.5$ then
    return $D + 1$
end if
if $f > 0.25$ then
    return $D/2 + 1$
end if
return $D/4 + 1$
```

Finally, the second variant, 'Classes, always cover' used the same four classes and parameters for the *class*() and *fill_class*() functions as the previous one. However, the *onCoverGeneration*() function is changed. If the number of received messages is lower than the current increase base $fw[j]$ for the flow j, it sends cover messages up to $fw[j]$. This can be presented with the pseudo-code from algorithm 6.

Algorithm 6. onCoverGeneration(i,j), add additional cover traffic to hide the temporary lack of traffic for the given round i and the flow j

```
$c = fw[j] - out[j][i]$
if $c \leq 0$ then
    return
end if
$out[j][i] = fw[j]$
```

6.1.3 The Algorithm of Zhu et al.

Zhu et al. [27] define the algorithm only for two sources and two destinations. We have extended it for more nodes so as to generate cover messages for all connections at the same time if they do not have any messages in the queue. If there is a message, it is sent. The authors suggested an extension by sending only on a limited number of connections, but did not provide an algorithm. We consider the choice a complicated problem, so we leave the algorithm in its base form.

6.1.4 The Algorithm of Shmatikov and Wang

The algorithm of Shmatikov and Wang [22] uses a traffic distribution with an average rate much higher than that in our simulation. Because of that, we have scaled the distribution so as to multiply each delay by ten. That yields an acceptable average inter-packet delay.

6.1.5 Result Processing

During the result analysis phase, we tried to correlate input and output flows. We calculated cross-correlations between the number of packets in each 0.1-second interval between the input and output links from each single simulation. We used the following equation:

$$r(d) = \frac{\sum_i [(x_i - mx) * (y_{i-d} - my)]}{\sqrt{\sum_i (x_i - mx)^2}\sqrt{\sum_i (y_{i-d} - my)^2}}, \tag{11}$$

where d is delay, r is cross–correlation, x_i and y_i are the signals, $i = 0, 1, ..N - 1$. mx and my are the mean values of the x and y signals, respectively.

Using the maximum $r(d)$ from each input-output pair, we calculated the best match for each input. Then we checked whether the best match was correct. We later refer to the ratio of correct matches as the detection rate.

6.2 *Results*

Figure 5 shows the rate of incorrect matches (source and destination not matched) for different algorithms as a function of flow length. The curve has a very similar shape for most of the algorithms: the detection rate is low for the shortest flow, then increases rapidly until the flow length reaches approximately 40 seconds. Then the rate still increases, but at a slower pace.

Unsurprisingly, the algorithm of Zhu et al. [27] gives the best protection, as it transforms each of the input flows into the same output. Full cover traffic with rounds yields similar results.

Of the other algorithms, the ones proposed in this work provide decent protection, above 70 per cent or above 40 per cent depending on the parameters, for the

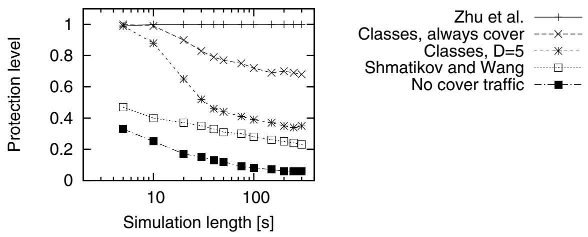

Fig. 5 Protection given by each of the simulated algorithms depending on the flow type, for all flows

longest flows. It should be noted, however, that there is a strong dependency on the simulation time. The shorter the simulation, the better protection is gained.

Surprisingly, the algorithm of Shmatikov and Wang did only a little better than the case with no protection at all. On investigation, we found out that it inherits too many of the original characteristics if the traffic distribution is not very similar to the assumed one.

6.2.1 Passive and Active Attacks

Figures 6, 7, 8 show the average protection rates for different traffic classes. The algorithm that transforms all flows to the same form shows similar results independently of the flow type.

The other algorithms clearly work better against passive attackers as shown in Fig. 6. The protection against active attackers is lower as shown in Fig. 7 and 8. It should be noted that the protection given by the algorithm of Shmatikov and Wang is very low under active attacks. Our algorithm provides significant protection for short observation lengths, up to approximately 50 sec (500 rounds).

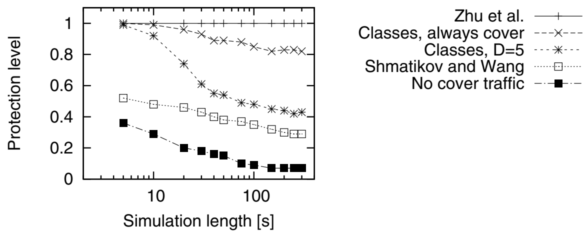

Fig. 6 Protection given by each of the simulated algorithms under passive attack conditions (flow types 1–4)

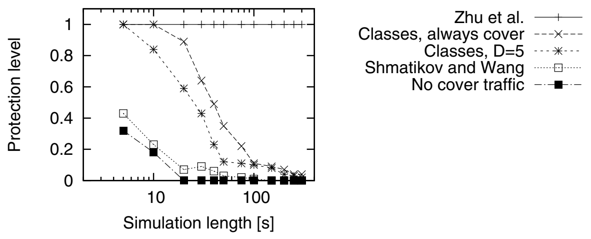

Fig. 7 Protection given by each of the simulated algorithms for flows with a single burst of traffic (flow type 6)

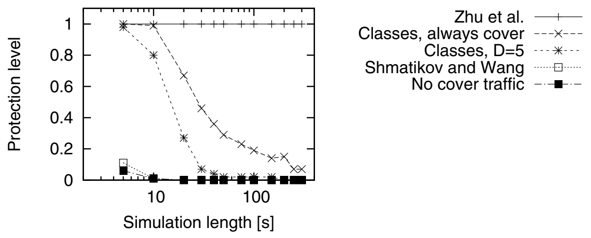

Fig. 8 Protection given by each of the simulated algorithms for flows with PN sequences (flow type 7)

6.2.2 Protection Cost

Detection ratio is an important factor when evaluating cover traffic algorithms, but it does not include all of the important aspects. The most important of the remaining ones is probably the cost of the algorithm, defined as the number of cover messages it must send.Table 1 shows the cover traffic ratio for different simulation lengths.

The higher cover ratio of our algorithm on short flows is caused by the initialisation of $in[j][i]$ to $B_{max}[j]$ (see 4.2). Other than that, values do not change significantly with simulation time. We can see that our algorithms provide significant reduction compared with Zhu and small reduction compared with Shmatikov and Wang. An ideal algorithm would have few cover messages and a low detection ratio (high

Table 1 Cover traffic ratio for different simulation lengths

Algorithm	Cover traffic ratio			
	T=5	T=20	T=100	T=300
Classes, $D = 5$ rounds	0.42	0.37	0.37	0.36
Classes, always cover	0.49	0.45	0.44	0.44
Shmatikov and Wang	0.45	0.45	0.45	0.45
Zhu et al.	0.60	0.60	0.60	0.60

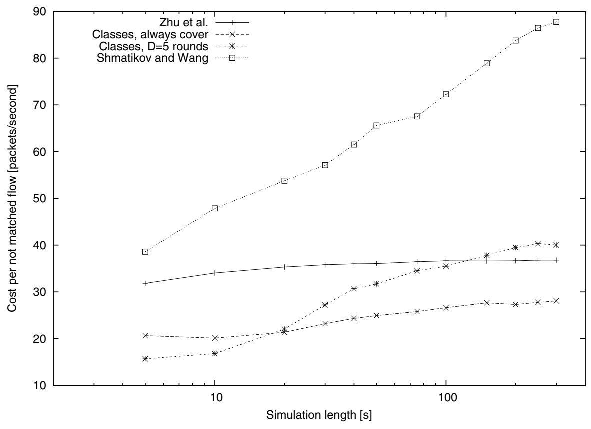

Fig. 9 Cost per each undetected flow for each of the simulated algorithms

protection). We have not found a comparison metric that incorporates both these aspects. Because of this, we propose a new metric, which we present below.

Using the same input traffic, we may compare different algorithms using the number of cover messages generated during a fixed interval T, divided by the number of successfully protected flows $N_{protected}$, where we define $N_{protected}$ as the number of flows not correlated by the attacker under certain attack scenerio. This can be presented as:

$$cost = \frac{\sum_{t=0}^{T_{total}-T} cover\_between(t, t+T)}{N_{protected} * T}. \tag{12}$$

Figure 9 shows the cost, as defined in (12), $cost_{T=1sec}$. It is worth noting that there are algorithms that scale well and those where the cost increases significantly when the simulation becomes longer.

Our algorithm has the lowest cost, but the set of parameters changes. For short flows, a small D is enough, but the cost of the algorithm with $D = 5$ increases, and even becomes higher than the cost of the algorithm of Zhu et al., for flows slightly longer than 100 seconds. For flows longer than 20 seconds, the second version of our algorithm has the lowest cost per undetected flow and has significantly lower cost than the algorithm of Zhu et al.

Shmatikov and Wang's algorithm in our simulations has a cost even higher than the algorithm of Zhu et al., and this cost increases quickly.

7 Conclusions and Future Work

We have shown that good protection against passive and short active attacks is possible with lower cost than in previous algorithms. Our class of algorithms addresses the basic conflict between protection and performance in the design of anonymity systems. Our simulations show that our algorithm uses less cover traffic than the algorithms that send the same traffic on all flows (36 to 44 per cent of the traffic for long simulations compared to 60 per cent needed by the algorithm of Zhu et al.). At the same time, the protection level is higher than in the algorithm of Shmatikov and Wang (40 to above 70 per cent compared to 20 per cent for long simulations).

Our algorithm provides decent protection against passive attacks. It also increases the time needed to perform an active attack. Further protection against active attacks may try to actively detect such attacks and perform countermeasures. The algorithms for such detection and the possible actions to perform are left for further study.

We have shown that our algorithm protects well against passive attacks, but less well against active attacks, especially long-lasting ones. However, our method leaves room for active attack detection. This can be done if the nodes transmit flow statistics to the initiator. These statistics would include the number of data and cover messages in each round. Based on those data, it should be possible to detect suspicious traffic patterns. That would include a situation when attackers have control over the first link and introduce delays.

The algorithm should be used by all intermediate nodes in the anonymity system. Then the traffic will be at least partially protected if the attackers introduce delay on certain links or have control over one of the intermediate nodes and those nodes fail to follow the prescribed cover traffic scheme.

We do not assume a specific transport protocol used in the anonymity system, and we do not address behaviour on losses. However, the protocol is optimised for datagram-based transport, with retransmissions handled (if necessary) at the higher (application) level. This is because we introduce strict restrictions on the flow. If used with a connection-based protocol (like TCP), the algorithm requires an implementation of flow control on the whole path.

Another parameter worth mentioning is the number of traffic classes. The number and behaviour of classes should be set based on the traffic in the specific anonymity system. Different extensions are possible here, keeping in mind that the more classes there are, the fewer flows there are in a single class, and the anonymity set becomes smaller. This requires planning, which may be done at the negotiation phase. Some balancing between classes may be desirable.

Our algorithm introduces latency. However, the latency is predictable. Additionally, we suggest the round time of 100 ms, what gives the total latency of 200 ms per hop. If that is too high, the round time may be shortened to 50 ms (100 ms per hop), or even further. We believe that such delay is acceptable, especially when the existing anonymity systems are reported to introduce latency measured in seconds [25]. Additional optimisations are possible, like processing the messages (especially encrypting/decrypting) during the latency period.

Finally, we also need new methods for evaluating anonymity in flow-based anonymity systems. Apart from the problems pointed by Tóth et al., the degree of anonymity metric which we use in this work also turned out to be complicated and time-consuming to calculate.

References

1. Adas, A.: Traffic Models in Broadband Networks. IEEE Communications Magazine 35(7), 82–89 (1997)
2. Back, A., Goldberg, I., Shostack, A.: Freedom Systems 2.1 Security Issues and Analysis. White paper, Zero Knowledge Systems, Inc. (May 2001)
3. Berthold, O., Federrath, H., Köpsell, S.: Web MIXes: A system for anonymous and unobservable Internet access. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
4. Chen, T.: Characterization of ATM on-off traffic from cell traffic measurement. In: 3rd IEEE International Workshop on Broadband Switching Systems (BSS 1999), pp. 53–57 (1999)
5. Dai, W.: PipeNet 1.1. Post to Cypherpunks mailing list (November 1998)
6. Danezis, G., Diaz, C.: A Survey of Anonymous Communication Channels. Technical Report MSR-TR-2008-35, Microsoft Research (January 2008)
7. Diaz, C., Preneel, B.: Taxonomy of Mixes and Dummy Traffic. In: Proceedings of I-NetSec 2004: 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems, Toulouse, France, pp. 215–230 (August 2004)
8. Diaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)
9. Dingledine, R., Mathewson, N., Syverson, P.: Tor: The Second-Generation Onion Router. In: Proceedings of the 13th USENIX Security Symposium, pp. 303–320 (August 2004)
10. Freedman, M.J., Morris, R.: Tarzan: A Peer-to-Peer Anonymizing Network Layer. In: Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002), Washington, DC, pp. 193–206 (November 2002)
11. Fu, X., Graham, B., Bettati, R., Zhao, W.: On Effectiveness of Link Padding for Statistical Traffic Analysis Attacks. In: ICDCS 2003: Proceedings of the 23rd International Conference on Distributed Computing Systems, Washington, DC, USA, p. 340. IEEE Computer Society, Los Alamitos (2003)
12. Kesdogan, D., Agrawal, D., Penz, S.: Limits of Anonymity in Open Environments. In: Petitcolas, F. (ed.) IH 2002. LNCS, vol. 2578, pp. 53–69. Springer, Heidelberg (2003)
13. Kesdogan, D., Egner, J., Büschkes, R.: Stop-and-Go MIXes: Providing Probabilistic Anonymity in an Open System. In: Aucsmith, D. (ed.) IH 1998. LNCS, vol. 1525, pp. 83–98. Springer, Heidelberg (1998)
14. Levine, B.N., Reiter, M.K., Wang, C., Wright, M.K.: Timing Attacks in Low-Latency Mix-Based Systems. In: Juels, A. (ed.) FC 2004. LNCS, vol. 3110, pp. 251–265. Springer, Heidelberg (2004)
15. ns 2 manual. Exponential on/off, `http://www.isi.edu/nsnam/ns/doc/node505.html` (accessed May 2009)
16. ns-2 webpage, `http://nsnam.isi.edu/nsnam/index.php/User_Information` (accessed: January 2009)
17. O'Connor, L.: Entropy Bounds for Traffic Confirmation. Technical Report 2008/365, IACR (October 2008)
18. Pfitzmann, A., Köhntopp, M.: Anonymity, Unobservability, and Pseudonymity – A Proposal for Terminology. In: Federrath, H. (ed.) Designing Privacy Enhancing Technologies. LNCS, vol. 2009, pp. 1–9. Springer, Heidelberg (2001), `http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.6.pdf`

19. Pfitzmann, A., Pfitzmann, B., Waidner, M.: ISDN-mixes: Untraceable communication with very small bandwidth overhead. In: Proceedings of the GI/ITG Conference on Communication in Distributed Systems, pp. 451–463 (February 1991)
20. Serjantov, A., Danezis, G.: Towards an Information Theoretic Metric for Anonymity. In: Dingledine, R., Syverson, P.F. (eds.) PET 2002. LNCS, vol. 2482, pp. 41–53. Springer, Heidelberg (2003)
21. Serjantov, A., Newman, R.E.: On the Anonymity of Timed Pool Mixes. In: Proceedings of the Workshop on Privacy and Anonymity Issues in Networked and Distributed Systems, Athens, Greece, pp. 427–434. Kluwer Academic Publishers, Dordrecht (2003)
22. Shmatikov, V., Wang, M.-H.: Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses. In: Gollmann, D., Meier, J., Sabelfeld, A. (eds.) ESORICS 2006. LNCS, vol. 4189, pp. 18–33. Springer, Heidelberg (2006)
23. Steinbrecher, S., Köpsell, S.: Modelling Unlinkability. In: Dingledine, R. (ed.) PET 2003. LNCS, vol. 2760, pp. 32–47. Springer, Heidelberg (2003)
24. Tóth, G., Hornák, Z., Vajda, F.: Measuring Anonymity Revisited. In: Liimatainen, S., Virtanen, T. (eds.) Proceedings of the Ninth Nordic Workshop on Secure IT Systems, Espoo, Finland, pp. 85–90 (November 2004)
25. Wendolsky, R., Herrmann, D., Federrath, H.: Performance Comparision of the low-latency Anonymisation Services from User Perspective. In: Borisov, N., Golle, P. (eds.) PET 2007. LNCS, vol. 4776, pp. 233–253. Springer, Heidelberg (2007)
26. Yu, W., Fu, X., Graham, S., Xuan, D., Zhao, W.: DSSS-Based Flow Marking Technique for Invisible Traceback. In: SP 2007: Proceedings of the 2007 IEEE Symposium on Security and Privacy, pp. 18–32. IEEE Computer Society, Los Alamitos (2007)
27. Zhu, Y., Fu, X., Graham, B., Bettati, R., Zhao, W.: On Flow Correlation Attacks and Countermeasures in Mix Networks. In: Martin, D., Serjantov, A. (eds.) PET 2004. LNCS, vol. 3424, pp. 207–225. Springer, Heidelberg (2005)