2009 International Conference on Intelligent Networking and Collaborative Systems

A Round-based Cover Traffic Algorithm for Anonymity Systems

Marta Rybczyńska
Institute of Telecommunications
Warsaw University of Technology
Warsaw, Poland
Email: marta@rybczynska.net

***Abstract*—In this paper 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. Simulation results show that our solution provides sufficient protection while reducing overhead traffic.**

***Index Terms*—Cover traffic, dummy messages, low-delay anonymity, privacy**

I. Introduction

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.

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 paper, 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 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 and attacks in Section II. We model our anonymity system and attackers in Section III. Section IV presents our algorithm. We start from basic concepts, then move to the core algorithm and possible extensions. In Section V, we present and discuss simulation results in terms of protection and cost. We conclude in Section VI.

This work was partially supported by Euro-NF.

II. Previous work

Anonymity systems have used cover traffic (also called 'dummy messages') as a protection method almost from the beginning. Notable examples include ISDN-Mixes [1], which use ISDN channels.

Cover traffic also appears in the earliest Internet-based designs, like PipeNet, where all clients transmit at the same rate [2]. 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 [4].

Fu, Graham, Bettati and Zhao [5] and Levine, Reiter, Wang and Wright [6] 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. [6] 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 paper.

In a Freedom system security analysis, Back, Goldberg and Shostack [7] 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 [8].

Zhu, Fu, Graham, Bettati and Zhao [9] 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 [10] 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 [11], 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.

III. 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 paper: 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.

IV. A class-based cover traffic algorithm

A. 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 paper 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.

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.

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.

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) 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.

B. 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 with the following pseudo-code (with the details of handling messages above the limit omitted for clarity's sake):

```
ONROUNDEND($i, j$)
1  $M[j] \leftarrow max(in[j][i - H_{len}]..in[j][i-1])$
2  if $M[j] <= B_{max}[j]$
3     then $out[j][i] \leftarrow class(M[j], j)$
4     else $out[j][i] \leftarrow class(B_{max}[j], j)$
```

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

C. 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 $[0, X)$.

The following pseudo-code describes our algorithm (with the details of handling messages above the limit omitted for clarity's sake, as in the previous version):

```
ONROUNDEND($i, j$)
1  $M[j] \leftarrow max(in[j][i - H_{len}]..in[j][i-1])$
2  $S \leftarrow 0$
3  for $k \leftarrow (i - H_{len})$ to $(i-1)$
4     do $S \leftarrow S + in[j][k]$
5  $f \leftarrow (S * 100)/(H_{len} * M[j])$
6  if $M[j] <= B_{max}[j]$
7     then $out[j][i] \leftarrow class(M[j])$
8     else $out[j][i] \leftarrow class(B_{max}[j])$
9  $fw[j] \leftarrow fill\_class(f, j)$

ONCOVERGENERATION($i, j$)
1  $c \leftarrow fw[j] - out[j][i]$
2  if $c <= 0$
3     then return
4  if $out[j][i] == 0$
5     then if $flow\_wait[j] > 0$
6              then $flow\_wait[j] \leftarrow flow\_wait[j] - 1$
7              else $flow\_wait[j] = 0$
8     else $flow\_wait[j] \leftarrow rand(fw[j]) + D/2 + 1$
9  $out[j][i] \leftarrow out[j][i] + c$
```

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

D. Discussion

The idea of rounds is similar to the mix time-out from timed mixes [12], 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 artefacts 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).

V. 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).

A. Implementation and test configuration

For evaluation, we have implemented and tested our cover traffic algorithms using the ns-2 simulator [13]. 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.

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 [11] 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.

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 following pseudo-code:

```
CLASS($N, j$)
1  if $N/B_{max}[j] > 0.75$
2     then return $B_{max}[j]$
3  if $N/B_{max}[j] > 0.5$
4     then return $3 * B_{max}[j]/4$
5  if $N/B_{max}[j] > 0.25$
6     then return $B_{max}[j]/2$
7  return $B_{max}[j]/4$
```

```
FILL_CLASS($f, j$)
1  if $f > 0.75$
2     then return $D * 2 + 1$
3  if $f > 0.5$
4     then return $D + 1$
5  if $f > 0.25$
6     then return $D/2 + 1$
7  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 following pseudo-code:

```
ONCOVERGENERATION($i, j$)
1  $c \leftarrow fw[j] - out[j][i]$
2  if $c <= 0$
3     then return
4  $out[j][i] = fw[j]$
```

3) The algorithm of Zhu et al.: Zhu et al. [9] 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.

4) The algorithm of Shmatikov and Wang: The algorithm of Shmatikov and Wang [10] 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.

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:

$$r(d) = \frac{\sum_i [(x_i - mx) * (y_{i-d} - my)]}{\sqrt{\sum_i (x_i - mx)^2}\sqrt{\sum_i (y_{i-d} - my)^2}}, \qquad (1)$$

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.

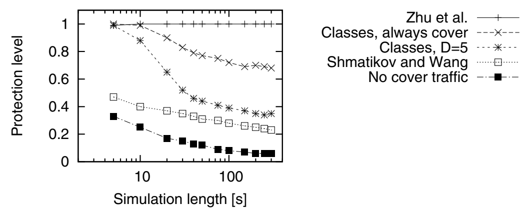
B. Results

Figure 1(a) 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.

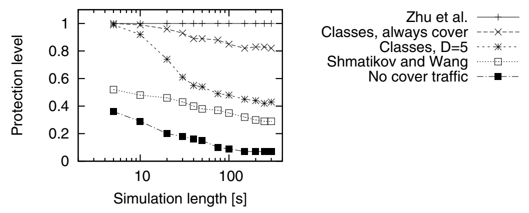
Unsurprisingly, the algorithm of Zhu et al. [9] gives the best protection, as it transforms each of the input flows into the same output. Full cover traffic with rounds yields similar results.

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

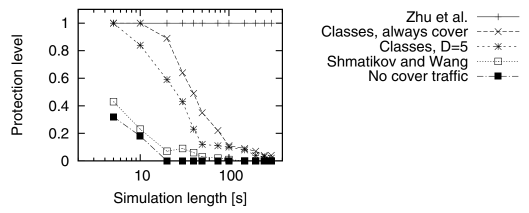
Surprisingly, the algorithm of Shmatikov and Wang did only a little better than the case with no protection at all. On investigation, we found out that it inherits too many of the original characteristics if the traffic distribution is not very similar to the assumed one.



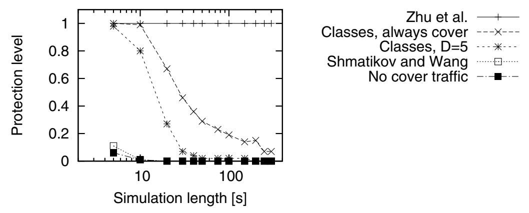
(a) All flows



(b) Flows under passive attacks (flow types 1–4)



(c) Single burst (flow type 6)



(d) PN sequences (flow type 7)

Fig. 1. Protection given by each of the simulated algorithms depending on the flow type.

1) Passive and active attacks: Figures 1(b), 1(c), 1(d) 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. 1(b). The protection against active attackers is lower as shown in Fig. 1(c) and 1(d). 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).

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.

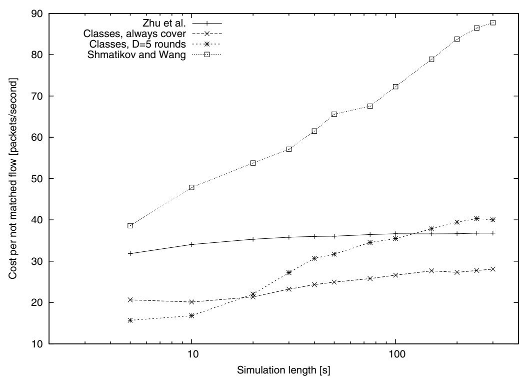


Fig. 2. Cost per each undetected flow for each of the simulated algorithms.

TABLE I
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

Table I 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 IV-B). 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 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:

$$cost = \frac{\sum_{t=0}^{T_{total}-T} cover\_between(t, t+T)}{N_{protected} * T}. \tag{2}$$

Figure 2 shows the cost, as defined in (2), $cost_{T=1sec}$. 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.

C. Discussion

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 [14]. Additional optimisations are possible, like processing the messages (especially encrypting/decrypting) during the latency period.

VI. 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.

We are currently implementing an anonymity system that uses our algorithm and uses information from the cover traffic algorithm to protect against not only passive attacks, but active attacks as well.

References

[1] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-mixes: Untraceable communication with very small bandwidth overhead," in *Proceedings of the GI/ITG Conference on Communication in Distributed Systems*, February 1991, pp. 451–463.

[2] W. Dai, "PipeNet 1.1," Post to Cypherpunks mailing list, November 1998.

[3] O. Berthold, H. Federrath, and S. Köpsell, "Web MIXes: A system for anonymous and unobservable Internet access," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 115–129.

[4] 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, pp. 193–206.

[5] X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Effectiveness of Link Padding for Statistical Traffic Analysis Attacks," in *ICDCS '03: Proceedings of the 23rd International Conference on Distributed Computing Systems*. Washington, DC, USA: IEEE Computer Society, 2003, p. 340.

[6] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing Attacks in Low-Latency Mix-Based Systems," in *Proceedings of Financial Cryptography (FC '04)*, A. Juels, Ed. Springer-Verlag, LNCS 3110, February 2004, pp. 251–265.

[7] A. Back, I. Goldberg, and A. Shostack, "Freedom Systems 2.1 Security Issues and Analysis," Zero Knowledge Systems, Inc., White Paper, May 2001.

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004, pp. 303–320.

[9] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On Flow Correlation Attacks and Countermeasures in Mix Networks," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004, pp. 207–225.

[10] V. Shmatikov and M.-H. Wang, "Timing Analysis in Low-Latency Mix Networks: Attacks and Defenses," in *Proceedings of ESORICS 2006*, ser. Lecture Notes in Computer Science, vol. 4189, September 2006, pp. 18–33.

[11] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "DSSS-Based Flow Marking Technique for Invisible Traceback," in *SP'07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*. IEEE Computer Society, 2007, pp. 18–32.

[12] C. Diaz and B. Preneel, "Taxonomy of Mixes and Dummy Traffic," in *Proceedings of I-NetSec04: 3rd Working Conference on Privacy and Anonymity in Networked and Distributed Systems*, Toulouse, France, August 2004, pp. 215–230.

[13] "ns-2 webpage," [Online], Accessed: January 2009, Available: http://nsnam.isi.edu/nsnam/index.php/User_Information.

[14] R. Wendolsky, D. Herrmann, and H. Federrath, "Performance Comparision of the low-latency Anonymisation Services from User Perspective," in *Proceedings of the Seventh Workshop on Privacy Enhancing Technologies (PET 2007)*, ser. Lecture Notes in Computer Science, vol. 4776. Springer, June 2007, pp. 233–253.