A Low-Variance Random-Walk Procedure to Provide Anonymity in Overlay Networks

J.P. Muñoz-Gea, J. Malgosa-Sanahuja, P. Manzanares-Lopez,
J.C. Sanchez-Aarnoutse, and J. Garcia-Haro

Department of Information Technologies and Communications
Polytechnic University of Cartagena
Campus Muralla del Mar, 30202, Cartagena, Spain
{juanp.gea,josem.malgosa,pilar.manzanares,
juanc.sanchez,joang.haro}@upct.es

Abstract. An alternative to guarantee anonymity in overlay networks may be achieved by building a multi-hop path between the origin and the destination. However, one hop in the overlay network can consist of multiple Internet Protocol (IP) hops. Therefore, the length of the overlay multi-hop path must be reduced in order to maintain a good balance between the cost and the benefit provided by the anonymity facility. Unfortunately, the simple Time-To-Live (TTL) algorithm cannot be directly applied here since its use could reveal valuable information to break anonymity. In this paper, a new mechanism which reduces the length of the overlay multi-hop paths is presented. The anonymity level is evaluated by means of simulation and good results are reported

1 Introduction

Over the last years, we have witnessed the emergence of different types of overlay networks in the Internet, such as the peer-to-peer file sharing systems or the real-time content delivery applications [1]. In these new scenarios, concerns about anonymity significantly arise among the user community. Anonymity refers to the ability to do something without revealing one's identity (in this case, the user's) [2]. The simplest solution to provide anonymity in overlay networks is to select several relay nodes in the route from the sender to the receiver. In this way, even if a local eavesdropper observes a message being sent by a particular user, it can never be sure whether the user is the current sender, or if the message is forwarded by a relay node.

Similar techniques have been widely studied in the past to provide anonymity in IP networks. One of them is Crowds [3], in which each node decides to deliver the message to a intermediate or destination node by flipping a biased coin (with probabilities p_f and $1-p_f$ respectively). Nevertheless, the use of this mechanism in overlay networks is not appropriate, because the forwarding procedure is not limited in any way, and as it known, in overlay networks neighbour nodes are connected by means of logical links, each one comprised of an arbitrary number of physical links. Fig. 1 shows an example of overlay network topology, which

S. Jajodia, and J. Lopez (Eds.): ESORICS 2008, LNCS 5283, pp. 238–250, 2008.

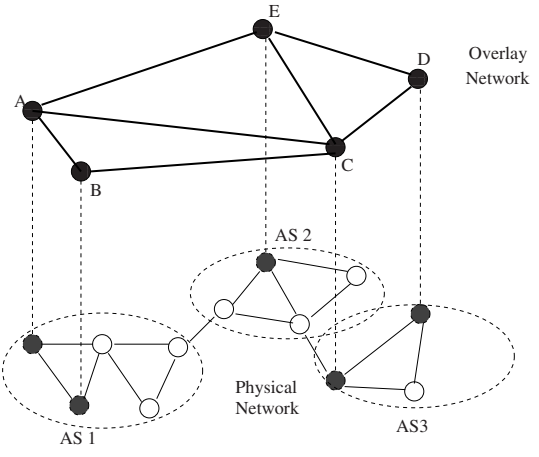

Fig. 1. Overlay Network

is composed of 5 overlay nodes from 3 ASes. From the figure, we can see that some of the overlay links are overlapped at physical layer even though they are completely disjoing at overlay service layer. This is one of the special characteristics of overlay networks. In addition, we can see that each of the overlay links is usually composed of several physical links [4]. Therefore, a serious increase in the length of the overlay path among the origin and the destination nodes could imply an exponential cost, in terms of bandwidht consumption and nodes overload.

A straightforward implementation to limit the path length makes use of the Time-To-Live (TTL) field, but there are multiple situations in which this implementation will immediately reveal to an "attacker" who the initiator node is. This paper proposes a mechanism that limits the length of overlay multi-hop paths without using a TTL (Time-To-Live) scheme. However, this mechanism is not restricted to this scenario, it can also be applied in a more general scenario. Furthermore, simulation results show that this mechanism presents a degree of anonymity equivalent to Crowds.

The remainder of the paper is organized as follows. Section 2 overviews some relevant works about anonymous systems. Section 3 presents the different requirements that must be satisfied in order to limit the path length. Section 4 introduces our proposal, called the *Always Down-or-Up* (ADU) mechanism. In section 5 our algorithm is evaluated analytically. In section 6 the anonymity level achieved by the proposed mechanism is evaluated by means of simulation. Finally, section 7 concludes the paper.

2 Related Work

The seminal paper on anonymous sytems was written by David Chaum [5]. He proposed a system for anonymous email based on the so called mix networks.

A mix node shuffles a batch of messages and delivers them in random order. The sender and the mix node use public key cryptography in order to hide the correspondence between input and output messages.

The mix networks design has been followed by many anonymous systems. The first widely used implementation of mix networks was the Type I cypherpunk anonymous remailers [6], using PGP [7] encryption to wrap email messages and deliver them anonymously. They were followed by MixMaster [8], and then MixMinion [9], which use the same basic principles, but split messages into equal-sized chunks and send each of them along potentially different routes, in order to defeat traffic analysis.

The concept of mix networks was first translated into the domain of general IP traffic by Wei Dalai, in his proposal for PipeNet [10]. PipeNet would build anonymous channels for low-latency, bidirectional communication, using layered encryption similar to Chaum's design. This layering suggested the title of Onion Routing for the first implementation of his type of IP forwarding [11]. Other implementations followed, including the commercial deployment of the Freedom Network [13] and the more recent effor behing Tor [12], a second-generation onion routing design.

Although the mix design has been quite influential, there are a number of notable alternatives. A network that uses a different approach is Crowds [3], designed for anonymous web browsing. Briefly, Crowds nodes forward web request to each other at random, executing a form of a random walk. At each step the random walk may probabilistically terminate and the current node then sends the request to the web server. The interesting feature of this system is that anonymity is achieved not only through having the messages under consideration forwarded by other honest nodes, but also through forwarding messages for other honest nodes and hiding the considered ones among them.

3 Background

In Crowds, the initiator node creates a packet containing a random path identifier, the IP address of the responder and the data. Then, it flips a biased coin. With probability $1-p_f$ (p_f is the probability of forwarding and it is a parameter of the system) it delivers the message directly to the responder or destination node, and with probability p_f it chooses randomly the next relay node. Each node receiving a packet with a new path identifier randomly decides- based on p_f- whether to forward it to the responder or to another (randomly chosen) relay node. With this original algorithm the forwarding procedure is not limited and, as we previously pointed out, it could be a tragedy regarding communication costs in an overlay scenario.

A possible solution is to restrict the maximum length of the paths. The system operates as the traditional scheme but, when the number of hops reaches a certain limit (called S), the path will be directed towards the destination node, irregardless of probability p_f. A straightforward implementation of the

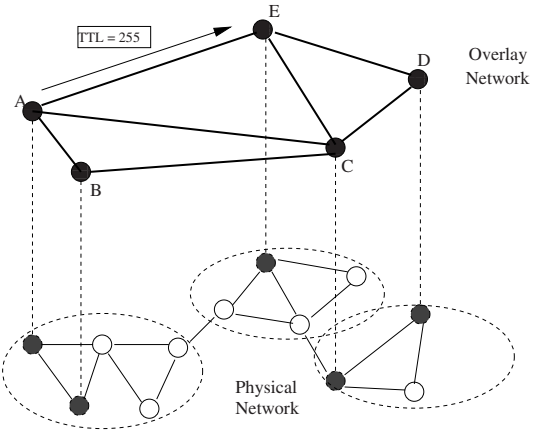

Fig. 2. Limitation of the TTL

bounded-length random walk consists of using a time-to-live (TTL) field, initially set to S, and processing it like in IPv4 networks [14]. In IPv4, TTL is an 8-bit field in the Internet Protocol (IP) header. The time to live value can be thought of as an upper bound on the time that an IP datagram can exist in an internet system. The TTL field is set by the sender of the datagram, and reduced by every router on the path to its destination. If the TTL field reaches zero before the datagram arrives at its destination, then it is discarded and an ICMP error datagram is sent back to the sender. The purpose of the TTL field is to avoid a situation in which an undeliverable datagram keeps circulating on Internet, and such a system eventually becoming swamped by such immortal datagrams. In theory, time to live is measured in seconds, although every host that passes the datagram must reduce the TTL by at least one unit. In practice, the TTL field is reduced by one on every hop. To reflect this practice, the field is named **hop limit** in IPv6.

However, there are multiple situations in which this implementation will immediately reveal to a "corrupt" node whether the predecessor node is the initiator or not. For example, in Fig. 2 we assume that the TTL has a predefined value of 255 for every path. In the example, node A is the path originator and randomly chooses node E as the next relay node in the path. Therefore, node A sends a packet to node E with a value of TTL = 255. If node E is a corrupt node, when it receives the packet it can easily deduce that node A is the path originator because the value of the TTL is the original. Therefore, with this simple solution the overlay network is not able to keep and adequate anonymity level.

An approach to solve this problem is to use high and randomly chosen (not previously known) values for the TTL field. However, the objective is to limit the forwarding procedure, and high values for the TTL represent long multi-hop paths. Therefore, the TTL would have to be small. But, in this case, the range of

possible random values for the TTL is too restricted, and it results in a similar situation to the one of a well-known TTL value among all the users. In this last case, corrupt nodes can easily derive whether the predecessor node is the origin of the intercepted message or not.

We can conclude that the TTL methodology is not appropriate to limit the length of multi-hop paths. Next section introduces a new mechanism that limits the length of overlay random walk paths without offering extra information to possible corrupt nodes.

4 Proposed Mechanism

The algorithm proposed in this work, as in Crowds, is based on the random-walk procedure. However, the variance associated to the length of the multi-hop paths is smaller than that in Crowds. Our objective is to limit the forwarding procedure. If the variance associated to the length of the paths is very high, it is possible that the real length of the path is also very high although the mean length of the path is not high. Our mechanism has a very low variance and it can be viewed as a quasi-deterministic mechanism of a statistical TTL implementation, because the real lenght of the path will be very similar to its statistical mean length.

Our first attempt is the *always-down* (AD) algorithm: The path originator chooses a uniform random integer (called u) between 1 and a predefined parameter M. If the value of u is equal to 1, the originator sends the request directly to the destination. Otherwise, the node forwards the request to a random node together with the random number u. The next node performs the same operation but replacing the upper bound M with the value of u. The mechanism continues in a recursive way, decreasing the size of the interval $[1, u)$ in each step. However, with this algorithm there is still correlation between the random number u and the hop length: although little values do not reveal anything about the path length, great ones do, since they can only appear at the first steps of the algorithm.

The opposite algorithm, called *always-up* (AU) has the same benefits and drawbacks. Now, at each step the node chooses a uniform random number between $(u, M]$. When a node selects M, the random walk procedure ends and the request is directly sent to the responder. In this case, great values of u do not reveal anything about the path length, but small ones do, since they can only appear at the first steps of the algorithm.

In order to avoid this critical issue, we propose to mix both mechanism as follows: The path originator chooses a random number (called u) between 1 and M. When this number is equal to 1 or equal to M, the originator node sends the request to the responder. If u is lower than a parameter LOW_BORDER, the algorithm works like AD. However, if u is greater than a paramater TOP_BORDER, the algorithm operates like AU. Finally, if u drops between LOW_BORDER and TOP_BORDER, the operation mode (AD or AU) is chosen randomly.

Fig. 3. Parameters of the algorithm

The parameters LOW_BORDER and TOP_BORDER are not fixed; every path originator chooses a random value for these parameters when it creates a new path. The only requirement is that these parameters are symmetric with respect to 1 and M.

This new algorithm is called *always down-or-up* (ADU) and it is able to statistically limit the length of the path in an anonymous environment. In order to speed up the algorithm, we introduce an additional parameter called e: If the new chosen random number is smaller than or equal to e (or it is greater than $M - e$) the originator node delivers the request to the responder.

The full set of parameters used by our algorithm is: M, e, LOW_BORDER and TOP_BORDER. Figure 3 represents these parameters in a numerical straight line.

5 Evaluation

Next, we present the analytical evaluation of the random variable l that represents the length of the path.

We define

$$P_{i,AD}(l = x) \tag{1}$$

as the probability that this random variable takes the value x in the AD algorithm with parameter $M = i$.

For the AD algorithm with parameters e and M we have deduced the following expressions

$$P_{M,AD}(l = 1) = \frac{e}{M} \tag{2}$$

$$P_{M,AD}(l = x | u_1 = i) = P_{i-1,AD}(l = x - 1) \quad e + (x - 1) \leq i \leq M \tag{3}$$

The interpretation of this first equation is obvious. On the other hand, Fig. 4 helps us to understand the second equation. This figure represents a specfic scenario with parameters $M = 5$ and $e = 2$. As can be observed, the calculation of every probability can be reduced to a smaller problem, in function of the previous probability and the first selected random number. The possible values for the first random number ($u_1 = i$) is restricted by the values of e and M, and also by the value of the probability to be calculated ($P(l = x)$). From this, we can obtain the possible values of the probabilities, that are also restricted

$$l \leq M - e + 1 \tag{4}$$

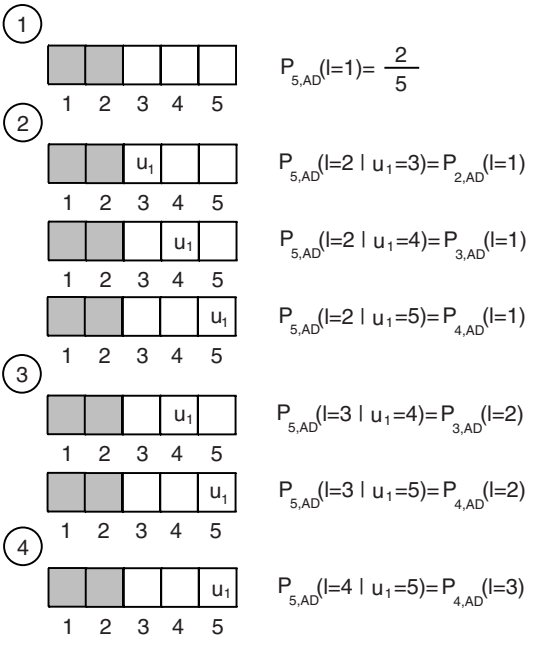

Fig. 4. Graphical interpretation

Next, we obtain a general expression for the previous equation

$$P_{M,AD}(l=1) = \frac{e}{M} \tag{5}$$

$$P_{M,AD}(l=x) = \frac{1}{M} \sum_{i=e+(x-1)}^{M} P_{i-1,AD}(l=x-1) \quad 1 < x \leq M-e+1 \tag{6}$$

As can be observed, the function obtained is a recursive equation. This means that the probabilities associated with values of l greater than 1 are calculated from the previous probabilities.

Next, we concentrate on the ADU algorithm. We know that

$$P_{M,ADU}(l=1) = \frac{2 \cdot e}{M} \tag{7}$$

In order to calculate the rest of probabilities, the problem can be reduced to three different sub-problems regarding the value of the first random number u. If it is lower than LOW_BORDER or it is greater than TOP_BORDER the algorithm works like AD or AU, respectively. The two previous scenarios can be interpreted as the AD algorithm when $M = LOW\_BORDER$, simplifying the study without loss of generality. However, in this case, the random variable

Table 1. Values of parameters for specific $\overline{l}$

	ADU		Crowds
$\overline{l}$	M	e	p_f
2	100	21	0.5
3	100	8	0.6667
4	100	3	0.75
5	150	2	0.80
6	350	2	0.8333

l is conditioned to be greater than 1, because to select this algorithm the path cannot finish in the first hop. Therefore

$$P_{M,AD}(l = x|x > 1) = P_{M,AU}(l = x|x > 1) = \frac{P_{M,AD}(l = x)}{P_{M,AD}(l > 1)} = \frac{P_{M,AD}(l = x)}{1 - \frac{e}{M}} \tag{8}$$

If the first random number u drops between LOW_BORDER and TOP_BORDER, the operation mode (AD or AU) is chosen randomly (RAND). In this case, without loss of generality, this scenario can be simplified as if the AD algorithm was always selected with $M = TOP\_BORDER$, since both AD and AU are equivalent in term of their statistical moments. Therefore, equation (6) can be used to find the probability, but now, the first selected random number has to be between LOW_BORDER and TOP_BORDER. Therefore

$$P_{M,RAND}(l = x) = \frac{1}{TOP\_BORDER} \sum_{i=LOW\_BORDER+(x-1)}^{TOP\_BORDER} P_{i-1,AD}(l = x - 1) \tag{9}$$

where the probability $P_{i-1,AD}(l = x - 1)$ is calculated using (6).

Finally, in this case the random variable l is also conditioned to be greater than 1, and therefore

$$P_{M,RAND}(l = x|x > 1) = \frac{P_{M,RAND}(l = x)}{1 - \frac{LOW\_BORDER}{M}} \tag{10}$$

The last step to calculate the probabilities associated with values of l greater than 1 is to appropriately weight the probabilities deduced for the three different possibilities (AD, AU and RAND):

$$\begin{aligned} P_{M,ADU}(l = x) = (1 - P_{M,ADU}(l = 1)) \cdot (P_{AD} \cdot P_{M,AD}(l = x|x > 1) + \\ + P_{RAND} \cdot P_{M,RAND}(l = x|x > 1) + P_{AU} \cdot P_{M,AU}(l = x|x > 1)) \end{aligned} \tag{11}$$

Then, the probability mass function can be used to calculate the associated statistical parameters. For example, to calculate the mean value,

$$E(l) = \overline{l} = \sum_{i=1}^{M} i \cdot P_{M,ADU}(l = i) \tag{12}$$

Table 2. Variance of the length of the paths

$\overline{l}$	ADU	Crowds
2	1.3337	2
3	2.3559	6
4	3.4135	12
5	4.5062	20
6	5.5821	30

Table 1 presents the appropriate values for the parameters M and e in order to achieve representative values for $\overline{l}$. The table represents low values (between 2 and 6) because, as we previously said, the objective is to achieve short multi-hop paths, which implies that the cost associated with the anonymous communication (bandwidth consumption and delay) may be reduced, and this type of application can be optimally implemented in overlay scenarios.

The previous table also represents the appropriate value of p_f to achieve the same values of $\overline{l}$ in Crowds. It is known that the mean length of the multi-hop paths created using the Crowds mechanism follows the geometrical expression: $\overline{l} = \frac{1}{1-p_f}$. We compare our scheme with Crowds because it is also based on random walk procedure.

Table 2 presents the variance of the length of the paths for ADU and Crowds. The variance of the ADU paths is calculated using:

$$V(l) = E(l^2) - E(l)^2 \tag{13}$$

on the other hand, for Crowds it is known that

$$V(l) = \frac{p_f}{(1-p_f)^2} \tag{14}$$

It is observed that the variance in ADU is always significantly smaller than in Crowds. This behaviour enables to interpret the ADU algorithm like a quasi-deterministic TTL implementation. Therefore, the mechanism achieves the target goal.

6 Analysis of Anonymity

The anonymity level achieved by the proposed mechanism is evaluated by means of simulation following the methodology exposed in [15]. This methodology is based on the use of the entropy as a measure of the anonymity level. This concept was presented in [16], and it can be summarized as follows: It is assumed that there is a total number of N nodes, C of them are corrupt and the rest honest. Corrupt nodes collaborate among them trying to find out who is the origin of the messages. Based on the information retrieved from corrupt nodes, the attacker

assigns a probability (p_i) of being the origin of a particular message for each node. The entropy of the system ($H(X)$) can be computed by:

$$H(X) = -\sum_{i=1}^{N} p_i log_2(p_i) \tag{15}$$

where X is a discrete random pariable with probability mass function $p_i = P(X = i)$. The degree of anonymity (d) of the system can be expressed by:

$$d = -\frac{1}{H_M} \sum_{i=1}^{N} p_i log_2(p_i) \tag{16}$$

where H_M is the maximum entropy of the system, which is satisfied when all honest nodes have the same probability of being the origin of the message.

If a message goes only through honest nodes the degree of anonymity will be $d|_{honest\ nodes} = d_h = 1$. If we assume that the message goes through at least one corrupt node with probability p_c and it crosses only through honest nodes with probability $p_h = 1 - p_c$, the mean degree of anonymity of the system is:

$$\overline{d} = p_c \cdot d|_{corrupt\ nodes} + p_h \cdot d_h = p_c \cdot d_c + p_h \tag{17}$$

Next, we present the simulation methodology proposed in [15]. First, let us introduce two random variables: X, modelling the senders ($X = 1, ..., N - C$), and Y, modelling the predecessor observed by the first corrupt node in the path ($Y = 1, ..., N - C, \oslash$; where $\oslash$ represents that there is not an attacker in the path). The probability distribution computed in equation 15 is really the distribution of X conditionated on a particular observation y. Therefore, the entropy metric evaluates $H(X|Y = y)$ for some y, and it can be calculated as

$$H(X|Y = y) = -\sum_{x} q_{x,y} log_2(q_{x,y}) \tag{18}$$

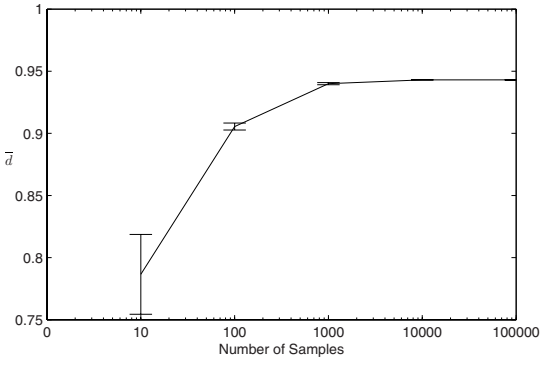

Fig. 5. $\overline{d}$ with confidence intervals as a function of the number of iterations

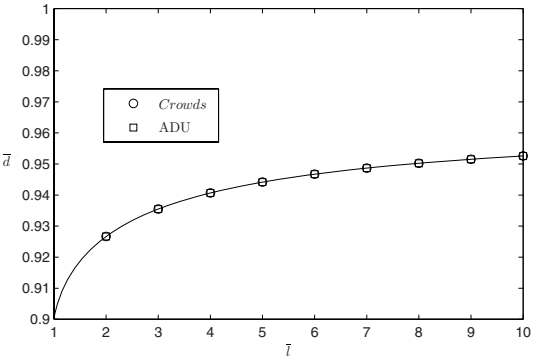

Fig. 6. $\overline{d}$ as a function of $\overline{l}$

The distribution $q_{x,y}$ is an estimate of the true distribution p_i. The way to calculate it is as follows: We keep a counter $C_{x,y}$ for each pair of x and y. We pick a random value x for X and simulate a request. The request is forwarded according to the algorithm until it either is sent to the destination or is intercepted by a corrupt node (this is 1 iteration). In the former case, we write down the observation $y = \oslash$, while in the latter case we set y to the identity of the predecessor node. In both cases, we increment $C_{x,y}$. This procedure is repeated a number of n iterations (later we will obtain an appropriate value for n).

Next, we can compute $q_{x,y}$ as follows

$$q_{x,y} = \frac{C_{x,y}}{K_y} \tag{19}$$

where $K_y = \sum_x C_{x,y}$.

On the other hand, the average entropy of the system, taking into account that a message can also go through honest nodes, can be expressed by

$$\begin{aligned} \overline{H} &= \sum_{y \geq 1} Pr[Y = y] H(X|Y = y) + \\ &+ Pr[Y = \oslash] H(X|Y = \oslash) = \sum_{y} Pr[Y = y] H(X|Y = y) \end{aligned} \tag{20}$$

With these definitions we can redefine expression 17 as,

$$\overline{d} = \frac{1}{H_M} \sum_{y} Pr[Y = y] H(X|Y = y) \tag{21}$$

The previous simulation methodology has been implemented in a simulator written in C language. First, in order to obtain an appropriate value for n we simulate an scenario with $\overline{l} = 4$ for different number of iterations. Figure 5 presents the results with the 95 % confidence intervals. We can see that from 10,000

iterations, the accuracy of the simulation results is within the 0.1 % with respect to the stable value. Therefore, in our simulations n is set to 10,000.

Figure 6 compares $\overline{d}$ for Crowds and ADU when N=100 and C=10. It represents the anonymity level according to the mean length of the paths, from 1 to 10. These small values have been selected because, as it was previously mentioned, our objective is to achieve shor multi-hop paths. It can be observed that the degree of anonymity achieved by the ADU algorithm perfectly matches the degree of anonymity achieved by Crowds.

7 Conclusions

In traditional (like Crowds) anonymous networks, the anonymity is achieved by building a multiple-hop path between the origin and the destination nodes. However, the cost associated with the communication increases dramatically as the number of hops also increases. Therefore, in these scenarios limiting the length of the paths is a key aspect of the protocols design.

Unfortunately, the common TTL methodology cannot be used to this purpose since corrupt nodes can employ this field to extract some information about the sender identity. Consequently, in this work an effetive mechanism to reduce the variance associated with the length of the random walks in anonymous overlay scenarios is proposed.

Our study reveals that the variance in ADU is always smaller than in Crowds. In addition, the degree of anonymity achieved by ADU is equivalent to Crowds. Thus, this mechanism is a recommended methodology to achieve a good trade-off between cost/benefit associated with the anonymity in overlay networks.

Acknowledgements

This research has been supported by project grant TEC2007-67966-C03-01/TCM (CON-PARTE-1) and it is also developed in the framework of "Programa de Ayudas a Grupos de Excelencia de la Región de Murcia, de la Fundación Séneca, Agencia de Ciencia y Tecnología de la RM (Plan Regional de Ciencia y Tecnología 2007/2010)". Juan Pedro Muñoz-Gea also thanks the Spanish MEC for a FPU (AP2006-01567) pre-doctoral fellowship.

References

1. Tsang, D.H.K., Ross, K.W., Rodriguez, P., Li, J., Karlsson, G.: Advances in peer-to-peer streaming systems [guest editorial]. IEEE Journal on Selected Areas in Communications 25(9), 1609–1611 (2007)
2. Pfitzmann, A., Hansen, M.: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management - a consolidated proposal for terminology version v0.30 (November 26, 2007) (2007), `http://dud.inf.tu-dresden.de/Anon_Terminology.shtml`

3. Reiter, M.K., Rubin, A.D.: Crowds: anonymity for web transactions. ACM Trans. Inf. Syst. Secur. 1(1), 66–92 (1998)
4. Li, Z., Mohapatra, P.: The impact of topology on overlay routing service. In: INFOCOM 2004: Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, Hong Kong, China, p. 418. IEEE Communications Society, Los Alamitos (2004)
5. Chaum, D.L.: Untraceable electronic mail, return addresses, and digital pseudonyms. Commun. ACM 24(2), 84–90 (1981)
6. Goldberg, I., Wagner, D., Brewer, E.: Privacy-enhancing technologies for the internet. In: COMPCON 1997: Proceedings of the 42nd IEEE International Computer Conference, Washington, DC, USA, p. 103. IEEE Computer Society, Los Alamitos (1997)
7. Zimmermann, P.R.: The official PGP user's guide. MIT Press, Cambridge (1995)
8. Möller, U., Cottrell, L., Palfrader, P., Sassaman, L.: Mixmaster protocol — version 2. draft (July 2003), `http://www.abditum.com/mixmaster-spec.txt`
9. Danezis, G., Dingledine, R., Mathewson, N.: Mixminion: Design of a type iii anonymous remailer protocol. In: SP 2003: Proceedings of the 2003 IEEE Symposium on Security and Privacy, Washington, DC, USA, p. 2. IEEE Computer Society, Los Alamitos (2003)
10. Dai, W.: Pipenet 1.1. Post to Cypherpunks mailing list (1998)
11. Syverson, P., Tsudik, G., Reed, M., Landwehr, C.: Towards an analysis of onion routing security. In: International workshop on Designing privacy enhancing technologies, pp. 96–114. Springer-Verlag New York, Inc., Heidelberg (2001)
12. Dingledine, R., Mathewson, N., Syverson, P.: Tor: the second-generation onion router. In: SSYM 2004: Proceedings of the 13th conference on USENIX Security Symposium, Berkeley, CA, USA, p. 21. USENIX Association (2004)
13. Back, A., Goldberg, I., Shostack, A.: Freedom systems 2.1 security issues and analysis. White paper, Zero Knowledge Systems, Inc. (2001)
14. Postel, J.: RFC 791: Internet Protocol (1981)
15. Borissov, N.: Anonymous routing in structured peer-to-peer overlays. Ph.D thesis, University of California at Berkeley, Berkeley, CA, USA. Chair-Eric A. Brewer (2005)
16. Díaz, C., Seys, S., Claessens, J., Preneel, B.: Towards measuring anonymity. In: Dingledine, R., Syverson, P. (eds.) PET 2002. LNCS, vol. 2482, pp. 54–68. Springer, Heidelberg (2003)