Tokens for anonymous Communications in the Internet

Arjan Durresi[1], Vamsi Paruchuri[1], Leonard Barolli[2], Raj Jain[3], Makoto Takizawa[4]
[1]*Department of Computer Science, Louisiana State University*
durresi@csc.lsu.edu
[2]*Department of Information and Communication Engineering, Fukuoka Institute of Technology*
barolli@fit.ac.jp
[3]*Department of Computer Science and Engineering, Washington University in St. Louis*
jain@cse.wustl.edu
[4]*Department of Computers and Systems Engineering, Tokyo Denki University*
taki@takilab.k.dendai.ac.jp

Abstract

With the growth and acceptance of the Internet, there has been increased interest in maintaining anonymity in the network. Using traffic analysis, it is possible to infer who is talking to whom over a public network. This work develops a novel approach to hide the senders and the receivers of messages. Routes are chosen and frames traverse these routes. Each frame consists of a token and a node can send a message through a frame only when the corresponding token is free. The best thing about our protocol is that it poses no bandwidth overhead when there is at least some traffic while posing minimal bandwidth overhead when there is no traffic at all.

1. Introduction

By using traffic analysis, it is possible to infer who is talking to whom over a public network. For example, in a packet switched network, packets have a header used for routing, and a payload that carries the data. The header, which must be visible to the network (and to observers of the network), reveals the source and destination of the packet. Even if the header were obscured in some way, the packet could still be tracked as it moves through the network. Encrypting the payload is similarly ineffective, because the goal of traffic analysis is to identify who is talking to whom and not (to identify directly) the content of that conversation.

The efficiencies of the public Internet are strong motivation for companies to use it, instead of private intranets. However, these companies may want to protect their interests. For example, a researcher using the World Wide Web (Web) may expect his particular focus to remain private, and inter-company collaborations should be confidential. Individuals may wish to protect their privacy as well. For example, the sending of e-mail should keep the identities of the sender and recipient hidden from observers. Also, a person shopping online may not want his visits tracked. Certainly someone spending anonymous e-cash would expect that the source of the e-cash be untraceable. The use of a packet switched public network should not require revealing who is talking to whom.

A purpose of traffic analysis is to reveal who is talking to whom. The mechanisms described here are designed to be resilient to traffic analysis i.e., to make it difficult for observers to learn identifying information about the connection. We present a novel approach to hide the senders and the receivers of messages. We use tokens for achieving anonymity. Routes are chosen and frames are scheduled to traverse these routes. Each frame is assigned a token and a node can send a message through a frame only if the token if free. The key advantage of the protocol is that it poses no bandwidth overhead when there is any traffic in the network, while posing minimal traffic when there is no traffic at all. We evaluate our protocol in terms of time complexity and communication complexity.

The remainder of this paper is organized as follows. Section 2 examines prior work in the areas of traffic analysis, privacy and anonymity; Section 3 defines the threat model for the system; Section 4 describes our protocol and its evaluation; Section 5 presents various enhancements and extensions; Section 6 presents some concluding remarks.

2. Related Work

In [5], Chaum describes a way to enable one participant to anonymously broadcast a message (DC-net). If the message is destined to a specific user, it can be encrypted with the user's public key. Since the message is received by all parties, recipient anonymity is trivially maintained. Unfortunately, this method has several serious drawback, among which that potential of Denial

Proceedings of the 17th International Conference on Database and Expert Systems Applications (DEXA'06)
0-7695-2641-1/06 $20.00 © 2006 IEEE

of Service, a participant can deny services to others by constantly sending messages through the DC-net. Improvement of DoS [2],[21],[22] still suffer from efficiency problems.

Secure multi-party computations are a related problem that has received considerable attention [6],[8],[9],[19]. A multi-party computation protocol can be used to hide participants' communication partners [17]. Multi-party computations that are secure in an asynchronous network are even more complex [4].

In [4], Chaum introduced the idea of the mix-net. Rackoff and Simon [17] define (and provide a proof of security for) a system that uses mix-nodes. Additional work has been done on mix-nets [7],[11],[12],[13],[14], [15]. In general, mix-nodes introduce some latency because messages are delayed by the mix, which can be acceptable for applications such as e-mail but less so for applications such as web surfing. On a more practical side, several systems providing fast, anonymous, interactive communication have been implemented. The first one was the Anonymizer [1] from Anonymizer.com. This solution offers rather weak security (no log safeguarding and a single point of vulnerability).

Crowds [18] consists of a number of network nodes that are run by the users of the system. Web requests are randomly chained through a number of them before being forwarded to the web server hosting the requested data. Crowds is also vulnerable to passive traffic analysis.

Onion Routing [10], [20] is another system that allows anonymous browsing. In this system, a user sends encrypted data to a network of so-called Onion Routers (essentially, these are real-time mixes). A trusted proxy chooses a series of these network nodes and opens a connection by sending a multiply encrypted data structure called an "onion" to the first of them. Each node removes one layer of encryption, which reveals parameters such as session keys, and forwards the encrypted remainder of the onion to the next network node. Once the connection is set up, an application specific proxy forwards HTTP data through the Onion Routing network to a responder proxy which establishes a connection with the web server the user wishes to use. The user's proxy multiply encrypts outgoing packets with the session keys it sent out in the setup phase; each node decrypts and forwards the packets, and encrypts and forwards packets that contain the server's response.

In spite of the similar design, Onion Routing cannot achieve the traffic analysis protection of an ideal mix-net due to the low-latency requirements [24]. The same is the case for the Freedom network as shown in [24]. In [16], the mix-net concept is extended to allow for interactive use in the special setting of digital telephony, while retaining most of its security features.

Beimel and Dolev in [23] describe a design inspired by the operation of a city bus. The protocol assumes all communicating nodes have a unique public/private key pair and know the unique predetermined circular path of the bus. The protocol also assumes a global clock with a global pulse such that a message can be delivered from one node to its neighboring node in one clock pulse.

3. The System and Thread Models

We consider a network of n processors, denoted $p1; : : : ; pn$, connected by m communication links. We use the communication graph $G(V;E)$ to represent our network, V is the set of processors and E is the set of communication links connecting the processors (that is $n = |V|$ and $m = |E|$). We assume that G is connected. Processors communicate by sending and receiving messages.

Some processor, *pi*, called *sender*, may decide to communicate with another (not necessarily neighboring) processor *pj* , called *receiver*. Our objective is to hide the fact that *pi* communicates with *pj* . That is, we want to hide the identities of *pi* and *pj*. Furthermore, our protocol even hides the fact that a message was sent. A protocol that achieves these goals is called an *anonymous message delivery protocol*.

We consider two types of adversaries *listening adversary* and *Byzantine adversary*. The listening adversary, also known as a honest-but-curious adversary, can monitor all the communication links of the network. We do not allow the adversary to monitor the internal contents of any processor. This adversary is honest, *i*.e., it cannot change any messages, delete messages, add any messages, or change the state of any processor.

The *Byzantine adversary* is more powerful than the listening adversary. The Byzantine adversary can monitor all the communication links of the processors of the network (except the memory of the sender and the receiver). In addition, for some parameter t, it can monitor and control up to t processors in the network. The *Byzantine adversary* if able to see the internal contents of these processors and also can insert messages, delete messages, through these monitors or arbitrarily change messages that they receive (before forwarding the messages). That is, these processors can deviate from the pre-defined protocol.

We evaluate a solution by its time complexity, its communication complexity, and its buffer complexity: the time complexity is the worst case time required to transmit a message from a sender to a receiver, the communication complexity is the maximal number of messages that are sent simultaneously by the processors in the network, and the buffer complexity is the buffer size required for each processor to store incoming and outgoing messages in each time step.

4. A simple token base Scheme for anonymous Communications

In this section we describe our scheme to achieve anonymous routing using tokens. In Section 5, we present various extensions and enhancements to generalize the idea of this protocol.

Notation: *Public and Private Keys*: We assume the presence of a Public Key Infrastructure. We denote the private and public keys of a node *i* as E_i and D_i. We denote the *E(M, k)* and *D(M, k)* to denote the encryption and decryption of message *M* with key *k*.

Network setup: We initially fix a spanning tree in the graph. Next, using an Euler tour (which is nothing but a DFS tour) of the spanning tree in the graph, we define a ring.

Tokens and Frames: At anytime there can be only one frame traversing through the ring[1]. The nodes use a *token passing access mechanism* to access a frame passing through the network. A node wishing to send data should first receive permission. When it gets control of the token, it may transmit data in that frame. Each *frame* is of fixed length and contains the *status* of the token itself. A token can be either in *free status* or *occupied status*. The format of the frame is as follows:

$<E((Token||E(Frame_{Header},E_d)||E(Frame_{Data},Ed)),E_i)>$

where E_i is the public key of the node *i*, that is upstream neighbor of sender and E_d is the public key of the destination.

The format of the Token is as follows:

< *Redundancy predicate* || *Status* >

Redundancy predicate is used for checking the validity of the frame. For the frame to be verified successfully by node *i*, upon decryption the *Redundancy predicate* must be fulfilled. *Status* specifies if the token is *occupied* or *free*. If a token is *free*, a node can send data through that frame; else it cannot. The format of the $Frame_{Header}$ is as follows:

< *Redundancy predicate* || *Source Address* || *Destination Address*>

Again *Redundancy predicate* is used for checking the validity of $E(Frame_{Header}, E_d)$.

The format of $Frame_{Data}$ is as follows:

<*Data length* || *Data* || *Padding*>

Data length specifies the length of the total *data* in the packet. This is crucial when the amount of data needed to be sent is not enough to fill the whole frame. In that case, data to be sent is padded with some random number to meet the constraint that the size of the frame is of fixed length.

[1] For simplicity and illustration purposes, we presently consider only one frame. Later in section 5, we present enhancements to deal with multiple frames.

4.1 The Protocol

Whenever a node *i* receives a frame, it decrypts the frame using its private key D_i and verifies the redundancy predicate. If the *Redundancy predicate* is not fulfilled, it reports an error to its downstream node (the node from which the frame was received). If the nodes were unable to recover the frame from the error, an error message is broadcasted and a new frame with a fresh token is generated. The claim process is detailed in Section 4.5. If the *Redundancy predicate* is fulfilled, then the following algorithm is executed.

1. If the node has no data to send, it just encrypts the resultant plain frame with its upstream node's public key and retransmits the packet on to the ring.
2. If the *status* of the token is *free* and the node has some data to send to another node D, then *i* constructs the frame as follows:
   - Node *i* constructs $Frame_{Header}$ and $Frame_{Data}$ as explained earlier using Destinations public key.
   - Node *i* sets the *status* field in the token to *occupied*.
   - Computes$<E((Token||E(Frame_{Header},E_d) ||E(Frame_{Data},Ed)),E_i)>$ using its upstream router's public key and transmits the packet on the ring.
3. If the *status* of the token is set to *occupied*, the node checks if the data in the frame is destined to itself by decrypting $E(Frame_{Header}, E_d)$ with its private key and checking if the *Redundancy predicate* is fulfilled.
   - If the frame is addressed to node *i*, then it makes a copy of it. Then it encrypts the whole frame with the public key of its upstream node and transmits the frame on to the ring.
   - Else, if the node is not the destination then the node just encrypts the whole frame with the public key of its upstream node and transmits the frame on to the ring.
4. Once the frame returns to the source, the source repeats the procedure as long as it has data to send. When it has no more data to send it sets the *status* field of the token to *free*, assigns the whole frame to some randomly generated data. Then it encrypts the token whole frame with its upstream router's public key and transmits the frame on the ring.

4.2 The claim process

Initially, when the ring is defined, a node is assigned an Active Monitor. One way of selecting an Active Monitor is to elect the node with highest MAC address as

the Active Monitor. An Active Monitor, which potentially can be any station on the network, performs a variety of ring-maintenance functions. One of these functions is the removal of continuously circulating frames from the ring. When a sending device fails, its frame may continue to circle the ring. This can prevent other stations from transmitting their own frames and essentially can lock up the network. The active monitor can detect such frames, remove them from the ring, and generate a new token. Another function of the active monitor is to generate a fresh frame when an error is detected in the circulating frame as explained in section

4.3. Protocol Evaluation

The communication complexity of the protocol is one and buffer complexity O(size of frame). The time complexity between two nodes is the distance between the nodes in the communication graph.

The protocol is very robust against both listening and byzantine adversaries. A listening adversary can monitor all the communication links of the network. Still, he would not be able to figure out the sender and destination pair because at node the frame is decrypted and encrypted and the frame is kept fixed in terms of length, thus giving no information to the adversary if the node is transmitting data or not. And as each node (including destination) retransmits the packet on the ring, the identity of destination is concealed. Even a Byzantine adversary cannot figure out the source-destination pair unless he is able to capture either of those nodes.

The protocol does not need high buffer space and the time complexity is also very good. But, the communication complexity is just one, which implies that a node cannot send more than one frame worth of data at a time and only one node can be sending at a time.

5. Enhancements

In this section, we present enhancements to achieve more efficiency in terms of communication complexity. That is more than one node can be communicating at a time. Also, we present some schemes that prevent any node from using up a frame all the time and also to achieve fairness among the nodes. That is, each node gets a fair chance to transmit the data.

5.1 Multiple Frames

In order to enable several nodes to be communicating simultaneously, we introduce multiple frames in ring, with each frame having its own token. This improves the efficiency of the network and also enables multiple communications to occur simultaneously.

The number of frames traversing the ring can be constant and this number can be obtained based on the amount of traffic in the network. Though, this is a simple method, this does not accommodate bursty traffic. So, instead we dynamically introduce (remove) frames into (from) the ring based on the ongoing traffic.

The monitor keeps a history of the ratio of number of *occupied frames* to the number of *free frames* for past T seconds. When this average goes above a threshold value, the monitor introduces new frames in to the ring. Also, a node can make an explicit request to the monitor for new frames. To enable this we introduce a new field – *have large data to send* – in to the *token*. A node when retransmitting a frame sets this field to one and then encrypts the frame with its upstream router's public key and then transmits the frame. Once the monitor gets a frame, it checks if any node has a lot of data to send and if some node indeed has, it introduces new frames.

Whenever the ratio of number of *occupied frames* to the number of *free frames* goes below a threshold, the monitor removes some frames from the ring. This ensures that the frames are nit unnecessarily transmitted in the ring.

5.2 Priority

We define a priority system that allows stations with high priority to use the network more frequently. The priority is defined by the frame's *priority* and *reservation* fields in the token. New format of the token is as shown below:

< *Redundancy predicate* || *Status* || *Priority* || *Reservation*>

Each node in the ring is assigned a priority level. In order to seize a token a station must have priority, which equals or is higher than the priority field of the token. Only then the station can reserve the token for the next pass around the network. This way when a node frees the token, the node with highest priority gets to capture the token. Nodes must change the priority back to its previous value after their transmission has completed.

6. Conclusion

In this paper, we presented a protocol for anonymous communications that is based on tokens. One advantage of our protocols over previous works is that they are not based on statistical properties for the communication pattern. Another advantage is that they do not require that all the processors in the communication network are busy. Our protocol poses no bandwidth overhead when there is

enough traffic while posing little overhead when there is no traffic at all or when the traffic is fluctuating a lot.

References

[1] anonymizer.com, The anonymizer.

[2] J. Bos and B.D. Boer, "Detection of disrupters in the DC protocol," In *Advances in Cryptology – EUROCRYPT '89* (1989), pp. 320–327.

[3] R. Canetti, "Studies in Secure Multiparty Computation and Applications*," PhD thesis, Department of Computer Science and Applied Mathematics*, The Weizmann Institute of Science, June 1995. revised version.

[4] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the Association for Computing Machinery,* 24, 2, Feb. 1981, pp. 84–88.

[5] D. Chaum, "The Dining Cryptographers Problem: Unconditional sender and recipient untraceability," *Journal of Cryptology,* 1, 1, 1988, pp. 65–75.

[6] R. Cramer, I. Damgard, Ard, S. Dziembowski, M. Hirt, T., Rabin, "Efficient multiparty computations with dishonest minority," In *Advances in Cryptology— EUROCRYPT 99,* March 1999, vol. 1561 of *Lecture Notes in Computer Science*, Springer- Verlag, pp. 311–326.

[7] Y. Desmedt and K. Kurosawa, "How to break a practical mix and design a new one," In *Advances in Cryptology – EUROCRYPT '2000* (2000), Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Heidelberg, pp. 557–572.

[8] R. Gennaro, M. O. Rabin, and T. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography," In *PODC: 17th ACM SIGACTSIGOPS Symposium on Principles of Distributed Computing,* 1998.

[9] O. Goldreich, S. Micali, and A. Wigdreson, "A. How to play any mental game—A completeness theorem for protocols with honest majority,". In *Proceedings of the nineteenth annual ACM Symposium on Theory of Computing, New York City, May 25–27, 1987,* New York, NY 10036, USA, 1987, ACM, Ed., ACM Press, pp. 218–229.

[10] D. Goldshlag, R. Reed, and P. Syverson, "Onion routing for anonymous and private internet connections," *Communications of the ACM (USA) 42*, 2, Feb. 1999, 39–41.

[11] M. Jakobsson, "Flash mixing," In *PODC: 18th ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing,* 1999.

[12] JAKOBSSON, M. A practical mix. *Lecture Notes in Computer Science 1403* (1998).

[13] JAKOBSSON, M., AND JUELS, A. Millimix: Mixing in small batches. Tech. Rep. 99-33, DIMACS, June 10 1999. Thu, 22 Jul 1999.

[14] OHKUBO, M., AND ABE, M. A length-invariant hybrid mix. In *Advances in Cryptology –ASIACRYPT '2000* (2000), Lecture Notes in Computer Science, International Association for Cryptologic Research, Springer-Verlag, Berlin Heidelberg, pp. 178–191.

[15] C. Park, K. Itoh, and K. Kurasawa, "Efficient anonymous channel and all/nothing election scheme," *Lecture Notes in Computer Science 765,* 1994.

[16] A. Pfitzmann, B. Pfitzmann, and M. Waidner, "ISDN-MIXes: untraceable communication with very small bandwidth overhead," In *Information Security, Proc. IFIP/Sec '91,* 1991, pp. 245–258.

[17] C. Rackoff and D. R. Simon, "Cryptographic defense against traffic analysis," In *Proceedings of the Twenty-Fifth Annual ACM Symposium on the Theory of Computing,* San Diego, California, 16–18 May 1993, pp. 672–681.

[18] M. K. Reiter and D. A. Rubin, "Anonymous Web transactions with crowds," *Communications of the ACM 42*, 2, Feb. 1999, pp. 32–48.

[19] A. Smith and A. Stiglic, "A. Multiparty computation unconditionally secure against adversary structures," *Cryptology SOCS-98.2*, School of Computer Science, McGill University, Montreal, Canada, 1998.

[20] P. F. Syverson, G. Tsudik, M. G. Reed, and C. E. Landwehr, "Towards an analysis of onion routing security," In *Proc. Workshop on Design Issues in Anonymity and Unobservability,* 25–26 July 2000, ICSI RR-00-011, pp. 83–100.

[21] M. Waidner, "Unconditional sender and recipient untraceability in spite of active attacks," In *Advances in Cryptology – EUROCRYPT ' 89* (1990), J.-J. Quisquater and J. Vandewalle, Eds., *Lecture Notes in Computer Science*, International Association for Cryptologic Research, Springer-Verlag, Berlin Germany, pp. 302–319.

[22] M. Waidner and B. Pfitzmann, "The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability," In *Advances in Cryptology—EUROCRYPT 89* (10–13 Apr. 1989), J.-J. Quisquater and J. Vandewalle, Eds., vol. 434 of *Lecture Notes in Computer Science*, Springer-Verlag, 1990, p. 690.

[23] A. Beimel and S. Dolev, "Buses for anonymous message delivery," In *Second International Conference on FUN with Algorithms*, Elba, Italy, May 2001, pp. 1–13.

[24] A.Back, U.Moller and A.Stiglic, "Traffic Analysis Attacks and Trade-Offs in Anonymity Providing Systems," In I.S.Moskowitz, editor, *IH 2001*, Volume 2137 of *Lecture Notes in Computer Science*, Springer-Verlag, 2001, pp 245-257.