A Practical Buses Protocol for Anonymous Internet Communication

Andreas Hirt Michael J. Jacobson, Jr. Carey Williamson
Department of Computer Science, University of Calgary
{hirt,jacobs,carey}@cpsc.ucalgary.ca

Abstract

This paper describes the design, implementation, analysis, and evaluation of a Practical Buses protocol for anonymous network communication. The protocol, based on the metaphor of a city bus, provides connection anonymity for Internet-based communication. We modify the original Buses protocol from the literature to provide receiver anonymity, strengthen security, and improve efficiency and scalability. Our prototype is tested in a LAN scenario. Preliminary performance results show that the Practical Buses protocol is promising.

Keywords: Anonymity, Privacy, P2P, Internet protocols

1. Introduction

The goal of anonymous communication is to protect the identities of communicating parties from eavesdroppers and adversaries. This facilitates freedom of speech without fear of retribution.

Two types of anonymity are required for complete anonymization [4]. *Data anonymity* filters out identifying data, such as the sender field in an e-mail. *Connection anonymity* obscures the communication patterns. In this paper, we focus on connection anonymity.

There are four types of connection anonymity. *Sender anonymity* [8] protects the identity of the initiator. *Receiver anonymity* [8] protects the identity of the responder. *Mutual anonymity* [5] provides both sender and receiver anonymity. *Unlinkability* [8] means that an attacker cannot discern sender-receiver relationships. Even if the identity of one endpoint is compromised, the identity of the other endpoint cannot be linked to it. Examples that require sender anonymity, receiver anonymity, mutual anonymity, and unlinkability are voting, publication of controversial documents, anonymous bulletin boards, and military communication, respectively.

In this paper, we present a practical Buses protocol that provides mutual anonymity from all perspectives, and is secure against all known attacks other than Denial of Service (DoS) attacks. Our protocol is based upon Buses [1], which uses the metaphor of a city bus. The bus hides a message's route through the network, much like a public transit bus hides a passenger's route through a city.

Our paper makes three main contributions. First, we identify and fix several weaknesses in the original Buses protocol. Second, we suggest several enhancements to facilitate a robust and efficient implementation of our protocol. Third, we present preliminary experimental results demonstrating the functionality and performance of our protocol. We believe that this is the first-ever implementation and evaluation of a Buses anonymity protocol. Preliminary experimental results show that the protocol is promising for LAN environments.

2. Background and Related Work

Classically, the three general techniques employed to provide anonymity are broadcasting [3], mixes [2], and re-routing [10]. Unfortunately, none of the foregoing approaches provides a perfect solution for anonymous Internet communication. Broadcasting and Mixes scale quadratically because they both require cover traffic to preserve anonymity. Re-routing techniques reduce cover traffic at the cost of being vulnerable to known attacks.

Buses [1] is a promising new anonymity technique that provides strong anonymity without the overhead of cover traffic. The optimal buffer complexity version of Buses is the only version that provides mutual anonymity. When the bus arrives at a node, the node replaces *every* seat on the bus with its decryption. Intelligible messages are removed and replaced with random data.

To send a message, a node first prepares a layered encryption of the message. It encrypts the message with the destination node's key, encrypts the encrypted message with the destination's predecessor's key, and so on until it encrypts the multiply-encrypted message with its successor's key. This layered encrypted message is placed in a random seat on the bus, and the bus is forwarded.

One layer of the layered encryption is removed at each hop in the tour of the network, until eventually the intended receiver removes the last layer of encryption and extracts the message. For reliable message delivery, the receiver returns an acknowledgment (ACK) in the same seat. Mutual anonymity is attained from everyone's perspective, including the receiver, because the layered encryption only allows a receiver to identify the predecessor in the bus route, not the original sender.

Although buses is similar to mixes because it uses layered encryption and a re-routing path (i.e., bus route), it differs in two important ways. First, buses do not require quadratic cover traffic to maintain strong anonymity. Buses has the bus sent between exactly two nodes every time unit. Second, buses requires *all* the nodes except the sender and receiver to be corrupted in order to defeat sender and receiver anonymity [6]. In mixes, only the re-routing nodes between the sender and receiver need to be corrupted in order to defeat anonymity.

3. Practical Buses Protocol

The Buses protocol is susceptible to replay attacks [6], bus size scales poorly, and many implementation-related issues are not addressed, such as acknowledgments, retransmissions, efficient encryption, and TCP/IP networking issues. The modified Buses protocol, which we refer to as the Practical Buses protocol, addresses these issues. Mechanisms are added[1] to improve scalability while preserving mutual anonymity. Features are also added to improve the security and efficiency of the protocol. For a complete description of the Practical Buses protocol, see [6].

3.1. Improving Scalability

The optimal buffer complexity Buses protocol scales poorly with a bus size of $O(ns^2)$ for moderate work loads ($s \geq \sqrt{n}$), where s is an upper bound on the total number of messages sent per bus tour and n is the number of participants. Our solution provides a tighter bound on bus size by using *owned seats*. Each participant exclusively owns k seats (e.g., $k = 2$) on the bus. The seats are identified by an owner field that contains the owner's public key. The resulting buffer size is $O(Kn)$, where K is an upper bound for k (e.g., $K = 4$), and $K << n$ for large n.

Each participant can use its owned seats to send and forward messages. A participant that receives the bus decrypts all of the seats, but does not replace them. Instead, it identifies valid seats to process, and only replaces its owned seats with any messages to forward or send, and if any excess messages exist they are queued for the next bus.

[1]For brevity reasons, the descriptions that follow assume a static network topology. However, dynamic network topologies can be handled using *p-threshold replacement back-off scheme* and *randomly delayed seat deletions* [6].

To determine if a decrypted seat is valid, the participant checks for fixed redundancy that is added to each nested layer before encryption. A flag is used to indicate whether the data obtained is a message intended for the participant, or a nested encrypted message that should be forwarded.

In addition, the overhead of layered encryption on the reverse of the re-routing path does not scale well. Our solution, which we call *nested encryption with indirection*, creates a random indirection path of random bounded length, with the nested encrypted message based on the reverse of that path. This allows the message to have fewer layers, reducing the overhead[2].

3.2. Maintaining Mutual Anonymity

Owned seats can compromise mutual anonymity. A participant can identify the sender of the message by monitoring who puts messages in their seats and when. To counter this threat, every time a participant receives a bus, it replaces all of its owned seats with messages to send, messages to forward, or random data. As long as an adversary cannot distinguish between random data and a valid encrypted message within polynomial time, mutual anonymity is maintained, even on low-traffic networks.

3.3. Improving Security

To improve security, the tampering of seats by anyone but its owner are detected by RSA-SSA signatures [12] that are appended to every seat. In addition, the loss of seats containing messages is detected by ACKs.

ACKs are incorporated by a sender adding a random 128-bit message tag and an anonymous public key in the innermost layer before encryption. After receiving a message, the receiver returns an anonymous ACK: an anonymous message sent with the sender's 128-bit message tag, a flag in the inner core indicating that it is an ACK, and an inner core that is encrypted with the anonymous public key.

An anonymous public key, rather than the public key in the seat owner field, is used to preserve sender anonymity. Otherwise, an adversary could monitor the network and correlate the anonymous public key to the participant that replaces seats that have the same key in the owner field.

3.4. Improving Performance

The Practical Buses protocol also improves performance by incorporating hybrid encryption, resends and acknowledgments, replay protection, expired seats, and persistent connections.

[2]This also supports a dynamic topology by providing a logical separation of indirection path from the physical routing path.

Hybrid encryption is used instead of a semantically secure cryptosystem because it is noticeably faster. In particular, a layer of the indirected encrypted message consists of a session key encrypted using the 1024-bit RSA-OAEP [11] public-key encryption scheme followed by the inner layer encrypted using the session key with 128-bit AES-CBC [7]. A salt field at the beginning of each encryption layer doubles as the AES key. As a result, a valid message is identified by having the prescribed OAEP redundancy.

Resends and ACKs are added to provide reliable message delivery. When a sender receives an ACK, it stops resending the message. If a sender does not receive an ACK within a timeout period, it re-sends the message via a nested encryption with a new indirection path and random AES keys. This creates a *disguised message* that is indistinguishable from its previous nested encryption. After a maximum re-send timeout, the protocol reports an error to the application to make the user aware that the message was not delivered.

Replay protection is added because the original Buses protocol is prone to replay attacks [6]. For each message that a participant receives, the decrypted inner layer as well as the value of a 128-bit salt field are recorded until a timeout elapses. The combination of the salt and message body uniquely identify a nested encrypted message. However, the receiver of a message records the message tag instead of the salt-field. This mechanism prevents a message from being received multiple times because it was resent multiple times by the sender due to an ACK timeout. To prevent replays after the timeout, a timestamp field is appended to the seat and the seat owner constructs a signature on the entire seat including the timestamp to prevent tampering. Received seats with a timestamp predating the replay protection timeout are deleted and ignored.

Expired seats are deleted to keep the bus size bounded. Each node checks for expired seats to delete by checking every seat's timestamp. A seat can expire when a participant unexpectedly disappears from the network without deleting its seats.

Persistent connections are used to make the connection setup, connection release, and TCP slow start a one-time cost. Persistent connections are important when propagation delay is significant, such as in a WAN.

4. Security Analysis

The analysis of the original Buses protocol in [1] uses a threat model that is composed of passive and active adversaries. A passive adversary is able to control one or more nodes, and see all the messages that pass through the node in addition to all of the node's local information such as its private keys. An active adversary has all the abilities of a passive adversary, but can also add, delete, or modify messages. The analysis in [1] concludes that Buses is secure under such a threat model. However, our work in [6] shows that it is vulnerable to a replay attack.

Due to the significant design changes to the Practical Buses protocol, it is necessary to re-evaluate its security against known attacks and new attacks. Our threat model also consists of a passive and active adversary, in addition to a global eavesdropper. A global eavesdropper can eavesdrop on all of the communication links used by the anonymous communication protocol.

The Practical Buses protocol is shown to resist all of the known attacks identified in [6]. This prior work categorizes anonymity attacks gathered from the literature, including local eavesdropper before/after proxy, size/time correlation, low load, marker, intersection, full compromised path, timing, passive traceback, replay, spam, mob, and filter attacks.

To identify new attacks, it is necessary to identify all of the observable events that an adversary could use to reduce anonymity. In our case, the attacker can observe seat replacements as a global eavesdropper or observe a valid message passing through a corrupted node.

Observing all the seat replacements does not reduce anonymity. All the seats are always replaced, so an attacker must be able to differentiate between a hybrid-encrypted message and random data. However, it is not known how to differentiate between random data and data using hybrid encryption with RSA-OAEP and AES-CBC. In fact, RSA-OAEP [11] [13] and AES [14] are designed so that their output passes vigorous statistical tests for randomness. Thus the replacement of a semantically secure cipher with hybrid-encryption does not reduce anonymity.

Observing a valid message pass through a node does not reduce anonymity. The attacker can only ascertain the predecessor of a valid message, but the successor could be anyone on the bus route. Furthermore, the predecessor of a message could have received the valid message to forward from any other node in the bus path. As a result, the identification of a predecessor as a sender requires that every other node be corrupted.

5. Experimental Evaluation

The primary performance metric for our prototype is *message latency*, defined as the elapsed time between when a user finishes entering a message and an acknowledgment of that message being received. The mean and standard deviation of the message latency are calculated over multiple messages read from a (repeatable) workload (e.g., 360 messages). Note that the message delivery time to the receiver is typically *half* of the message latency.

The LAN tests used a Beowulf Cluster of 14 identically-configured dual-processor 2.4 GHz Pentium machines at the University of Calgary. Each machine has 2GB RAM and a

512 KB cache. They are connected with a dedicated 1 Gbps Ethernet switch. The software environment is Linux 2.4.20-19.7, POSIX threads with libc-2-2.5, GMP 4.1.1, and g++ 3.2 To simplify the analysis of the results, a single processor is used on each machine.

Our initial experiments studied the effect of various factors on average message latency. In [6] it is shown that message latency is independent of message size, but strongly dependent upon the seat size, number of network nodes, and the number of indirection layers used. The protocol performs well under random sender-receiver traffic and Poisson message arrival process. For space reasons, only the results from one LAN test are shown here (see Figure 1). This experiment used a Poisson message arrival process with an aggregate rate of 30 messages per minute. Message traffic is non-uniformly distributed among the nodes in the network with a Zipf-like distribution. Message sizes are random, ranging in size from 1 byte to 2 KB. The seat size is 3 KB and the number of indirection layers is randomly chosen as either 1 or 2.

Figure 1 plots the average message latency as the number of nodes is scaled from 2 to 14. The vertical dotted lines show one standard deviation above and below the mean message latency. Figure 1 highlights the scalability characteristics, showing approximately linear growth of the average message latency. The protocol is also fair, as shown in [6].

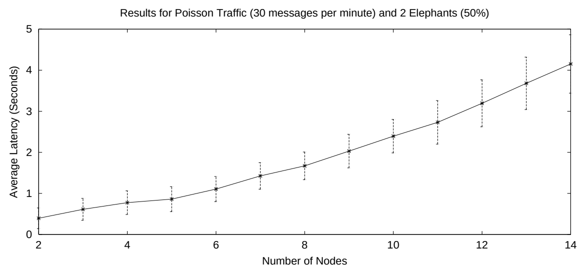

Figure 1. results for uniform random traffic

6. Conclusions

The design goals for our Practical Buses protocol include very strong anonymity, medium-latency, and support to be extended to a P2P design as well as a dynamic network topology. The novel techniques central to this design are owned seats and nested encryption with indirection.

The Practical Buses protocol performs reasonably in a moderately-sized static network, under various traffic patterns. The send-ACK latency for messages on a LAN are typically 1-4 seconds, while the send-receive latency is typically half of that.

Future research directions include extending the testing to a larger WAN testbed such as PlanetLab [9] and the derivation of a model. The performance could also be improved by adding in symmetric key tunneling and multiple bus routes. The former should reduce the constant costs, and the latter should produce logarithmic scalability. Ideally, this would create the first strong anonymous communication scheme with a low-latency overhead.

References

[1] A. Beimel and S. Dolev. Buses for Anonymous Message Delivery. *Journal of Cryptology*, 16(1):25–39, 2003.

[2] D. Chaum. Untraceable Electronic Mail, Return Addresses and Digital Pseudonyms. *Communications of the ACM*, 24(2):84–88, 1981.

[3] D. Chaum. The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability. *Journal of Cryptology*, 1(1):65–75, 1988.

[4] J. Claessens, B. Preneel, and J. Vandewalle. Solutions for Anonymous Communication on the Internet. In *Proceedings of the International Carnahan Conference on Security Technology*, pages 298–303. IEEE, 1999.

[5] Y. Guan, X. Fu, R. Bettati, and W. Zhoa. An Optimal Strategy for Anonymous Communication Protocols. In *Proceedings of 22nd International Conference on Distributed Computing Systems*, pages 257–266. IEEE, 2002.

[6] A. Hirt. A Practical Buses Protocol for Anonymous Network Communication. Master's thesis, University of Calgary, Calgary, Alberta, August 2004.

[7] National Institute of Standards and Technology. *Advanced Encryption Standard — FIPS 197*, 2001. http://csrc.nist.gov /publications/fips/fips197/fips-197.pdf.

[8] A. Pfitzmann and M. Waidner. Networks without User Observability. *Computers & Security*, 2(6):158–166, 1987.

[9] PlanetLab, 2005. `http://www.planet- lab.org` .

[10] M. Reiter and A. Rubin. Crowds: Anonymity for Web Transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[11] RSA Laboratories. *RSA-OAEP Encryption Scheme — Algorithm Specification and Supporting Documentation*. RSA Security Inc, 2000. ftp://ftp.rsasecurity.com /pub/rsalabs/rsa_algorithm/rsa-oaep_spec.pdf.

[12] RSA Laboratories. *RSA Signature Scheme with Appendix — Probabilistic Signature Scheme*. RSA Security Inc, 2000. ftp://ftp.rsasecurity.com /pub/rsalabs/rsa_algorithm/nessie_pss.zip.

[13] RSA Laboratories. Recent Results on OAEP Security, 2004. `http://www.rsasecurity.com/rsalabs/ node.asp?id=2147$\#$FOPS` .

[14] J. Sotto and L. Bassaham. Randomness Testing of the Advanced Encryption Standard Finalist Candidates. Technical report, National Institute of Standards and Technologies, 2000.