Secure Anonymous Acknowledgments in a Delay-Tolerant Network

Edoardo Biagioni
University of Hawai'i at Mānoa
esb@hawaii.edu

ABSTRACT

***Abstract*—Acknowledgments are used in TCP and other protocols for reliable transmission of data. This paper describes a special kind of acknowledgment with additional features, including in particular a guarantee that only the intended receiver can issue a valid acknowledgment for a given message. Unlike TCP, these acknowledgements work well even for connectionless communications.**

This acknowledgement mechanism assumes that secure encryption is used to protect message data. The sender of a data message includes a randomly-generated acknowledgement in the encrypted part of the message. As long as only the intended receiver can decrypt the message, only the intended receiver can issue the acknowledgement when it receives the message. Such randomly-generated acknowledgments in no way identify senders and receivers, providing a degree of anonymity.

In a Delay-Tolerant Network all hosts participate in forwarding data, taking on the role that routers fill in a more conventional network. Hosts cache messages for later delivery to other hosts that may be connected only intermittently. To keep caches manageable, messages must expire after a certain time.

To further improve cache management, the cryptographic hash of each message acknowledgement is included as the message identifier in the unencrypted part of the message. A forwarding host that receives an acknowledgement can hash it and remove from its cache any message matching this message identifier, and decline to forward any future such messages.

I. INTRODUCTION

Few features of networking protocols are as familiar to networking professionals as the acknowledgement, commonly abbreviated **ack**. TCP connections rely on acks to confirm receipt of transmitted data and to grant permission to send further data. This makes acks essential to the reliable transmission provided by TCP. Each TCP ack carries a 32-bit ack number reflecting the sequence number of the next byte of data expected on the connection.

TCP acks are very predictable, which makes it easy for an attacker to send spoofed acks [1].

The contribution of this paper is an ack mechanism that provides:

- a guarantee of delivery, since valid acks can only be issued by the intended receiver as identified by that receiver's access to a specific secret key (Sections III and IV).
- a measure of anonymity, in that the ack message in no way identifies either the sending or the receiving device (Section III).
- the ability to reliably and securely delete acked messages from message caches, even on hosts that have no access to any secret key (Section III).
- the ability for a receiver to communicate back to the sender that partial receipt of a message is sufficient (Section V).

II. BACKGROUND

A. Delay Tolerant Networks

Delay-Tolerant Network (DTN) technologies [2][3][4] support communication among hosts that may only occasionally be connected to each other. Such conditions are common among mobile hosts that communicate through an ad-hoc network.[1] The purpose of DTNs is to deliver data even in the absence of any simultaneous end-to-end path between sender and receiver.

[1]In an ad-hoc network, every host may contribute to both data forwarding and to creating and receiving data.

In the 1970s and 1980s email was often delivered even to hosts that were never directly connected to the Internet [5]. Such a host H would from time to time dial up another host G (Gateway) with better connectivity and which had agreed to cache email to and from H. The connection used a protocol called UUCP (unix-to-unix copy) [6] to download emails addressed to the users of H, and to upload emails originated by users of H.

To support this intermittent email delivery, G had to save messages addressed to users of H and deliver them on request, and buffer outgoing messages originating from H until G itself could in turn connect to its upstream host or directly to the wider Internet to deliver these messages. These techniques allowed email to be delivered even if neither sender nor receiver of the email were ever directly connected to the Internet.

DTNs have similar goals as the old uucp email system, but delivering general-purpose messages rather than just email. Intermediate hosts in a DTN cache messages and deliver them on request. Sometimes the intermittent connection may be established when one of the hosts moves into wireless range of another host.

To accomplish this delivery, in a DTN all hosts are peers, so all hosts must include all the functionality that in a more conventional network is divided among data sources, data sinks, and routers.

When DTN nodes have access to the Internet, messages can be delivered directly from the sender to the receiver. The interesting case is therefore when either the sender or the receiver, or both, are offline and not communicating over the Internet. Such disconnected hosts communicate directly with each other over ad-hoc links, and send messages to all peers. As connectivity changes to allow communication with new nodes, peers can forward their cached messages to any new peers, with the goal of eventually delivering each message to its final destination.

B. A Useful Delay Tolerant Network: AllNet

One application of DTNs is chat or text message delivery among mobile devices with insufficient data connectivity to the Internet. Mobile devices are widespread even where connectivity is intermittent, especially in rural areas or wherever a mobile device is outside the coverage of its wireless provider for any reason, including wilderness adventure, foreign travel, and in some emergency situations.

The AllNet project [7] is designed to create such a DTN among mobile devices. AllNet is designed to work whenever devices can communicate directly among each other even in the absence of cellular service. Available technologies include point-to-point (infrastructure-less) WiFi and the many variants of Bluetooth, particularly Bluetooth Low Energy (BLE). All of these are available on popular mobile devices, though in each case the operating system typically imposes idiosyncratic restrictions on their usage.

AllNet has been designed to provide some degree of security and anonymity of communications, including features such as:

- connectionless communication which does not need to identify sender and receiver.
- pervasive encryption of data messages.
- optional addresses that are unencrypted, and only used to provide improved performance.
- support for ad-hoc communications over direct (non-Internet) links between hosts.
- direct key exchanges between any two parties in a communication, without involving third parties.
- protection against DDoS amplification attacks [8].

III. Acknowledgements in an Ad-Hoc Network

Since DTNs require all-to-all delivery when in ad-hoc mode, conceptually each host in a DTN has to cache all messages. While the storage requirements for a chat application are moderate, exchanging this data with every peer that one encounters may require substantial spectrum and too much energy from a limited battery. Instead, AllNet acknowledges data messages that have been received by the intended receiver by sending a small ack message that confirms receipt. This is conceptually similar to a TCP ack, but with the following major differences:

- receipt of an ack confirms message delivery to the application, not just the transport layer.
- only the intended receiver of the message can issue the ack.

- acks are 16-byte random strings that are almost certainly unique to each message.[2]
- nothing in the ack explicitly identifies either the sender or the receiver, so these acks are anonymous.
- several such acks from different conversations can be combined into a single ack message.

When a sender receives an ack for a message it has sent, both its cache and the application can record that the message has been delivered to the intended device.

Equally important, other hosts in the DTN can record that the message has been delivered and evict it from their caches, since any further delivery would not serve any useful purpose.

Ack messages are forwarded in the same way as other message, with each ack message possibly multiple individual acks. This means that ack delivery has the same issues as delivering messages in the first place, but acks can be much smaller than data messages, so the overhead of distributing them is less.

Further, acks are ephemeral, which means that they can be deleted without affecting data transmission – failure to deliver or cache an ack may result in duplicate message transmission, which is a minor consequence compared to failure to deliver a data message.

Acks in most systems are idempotent, meaning that receiving the same ack once has the same effect as receiving it multiple times. Therefore, in TCP as in AllNet, duplicate transmission of acks has no consequences beyond the cost of transmission.

IV. Secure Acknowledgements

In a network using TCP it is easy for an attacker to generate acks identical to those that would be generated by an intended receiver. [1]. As a result TCP is not secure, and any guarantee of delivery must be provided by a higher layer such as TLS.

AllNet is designed to provide many of the features of TCP and TLS for connectionless decentralized ad-hoc networks. For example, instead of the hierarchically issued certificates used in TLS, AllNet certifies keys based on interpersonal interactions among users. The security of these transmissions can only be guaranteed if the intended receiver of a message is the only system able issue the corresponding ack.

In AllNet, each receiver holds one or more cryptographic private keys that it uses to decrypt messages addressed to itself.

A sender generating a data message for a specific receiver then includes the ack in the message before encrypting it, so that the ack is included in the encrypted part of the message.

The sender then adds to the unencrypted part of the data message the cryptographic hash of the ack.[3] This hash is known as the message identifier or **message ID**. The message ID, like the ack, is very likely to be unique for each message.

Every host forwarding the message can see the message ID, but is unable to generate a valid ack. Generating a valid ack would require either obtaining the receiver's key or inverting the cryptographic hash function. Inverting the hash function is assumed to be difficult when using a strong cryptographic hash function such as the SHA-512 hash [9] used by AllNet.

On the other hand every host receiving a new acknowledgement can hash it and remove from its cache any messages with a matching message ID.

A. Anonymity of Messages and Acknowledgements

AllNet has optional addressing. Each source and destination address has a number of significant bits. A sender desiring anonymity may send its message with 0 bits of source and destination address, so that no unencrypted addresses are visible in the message. When addresses have 0 bits, they identify every host on the network, and every host receiving the message may then attempt to decrypt every such packet. A successful decryption then means the message is for this host. Because decryption requires resource usage from all the hosts in the network that match a given address pair, AllNet assigns lower priority to messages with fewer address bits than to messages that have more bits specified in the address.

[2]The chances of a collision among randomly generated 128-bit strings is small as long as the number of such strings is substantially less than $2^{64} = 18,446,744,073,709,551,616$.

[3]Normally such hashes produce more than 16 bytes of data, so the sender only includes the first 16 bytes of the hash.

Just as messages can be anonymous, so acks in AllNet are also anonymous in that the ack itself carries no information about the sender and receiver of the ack. In addition, since any host in the network might have cached the corresponding message, to the extent possible acks are distributed to every host in the network. This universal distribution makes acks more resistant to traffic analysis than if they were only delivered to the original sender.

V. Acknowledgements for Messages Larger than the Maximum Transmission Unit (MTU)

These secure acknowledgments provide interesting options when a message exceeds a network's Maximum Transmission Unit (MTU) and has to be sent as a number of smaller packets. Such large messages occur when sending multimedia data such as audio, images, and video.

In the Internet Protocol (IP) uses a mechanism called **fragmentation** [10], and this paper uses the same term. The receiver can reconstruct the larger message once it has received all of the fragments.

Each fragment in AllNet carries two acks, one for the message as a whole and the other for the specific fragment. Correspondingly, the unencrypted part of each fragment contains a message ID obtained from hashing the message ack, and a fragment ID (which AllNet calls a packet ID) obtained from hashing the fragment ack.

A receiver receiving fragments of a larger message may ack them individually. Once the receiver has received all the fragments of a message, it then issues the ack for the entire message. Each host receiving a message ack can clear from its cache every fragment of the larger message, even if it is missing one or more individual fragment acks.

A. Acknowledging Partial Transmission

It is interesting to speculate on different ways of using these message and fragment acks. In particular, it may be that only part of a message is needed to deliver the desired content.

For example, the text of many email messages is sent in duplicate, once as plain text and once as html. If this strategy were used by an AllNet client, then a receiver that received enough fragments to reconstruct one or the other alternative could immediately issue the message ack without requiring retransmission of the missing fragments.

Similar techniques could apply when sending images or videos. Many devices have screens with low resolution and do not benefit from displaying high resolution images. Other devices have screens with very high resolution, and their users would be dissatisfied with low resolution images. A clever use of this protocol might send the low resolution image first. If the sender receives acks for all of the fragments, but not for the message, it can then send the high resolution image. Conversely, if it receives a message ack without sending the high resolution image, it need not send the high resolution image at all.

A further intriguing idea is the use of forward error correction, that is, the transmission of data beyond the minimum required, to achieve reliable transmission even in the face of loss of some of the fragments, and with no need for retransmission. Avoiding the need for retransmission can be particularly useful for delay-tolerant networks.

To explain, we refer to a particularly simple scheme for forward error correction, in which each fragment of data is transmitted three times. In this triply-redundant scheme, the message ack and message ID are the same for all the fragments. Each of the three copies of each fragment of data carries the same content but different fragment acks and fragment IDs.

With this algorithm, at least one copy of the data is likely to be delivered even in the face of loss of a small number of the fragments. A receiver that can reconstruct the original message can immediately issue the message ack, even though some of the fragments may never be delivered.

If synchronous communication is available, the sender may receive the message ack for such a triply-redundant transmission before sending three copies of all the fragments. On the other hand in a DTN, a receiver may over time receive a random subset of the fragments, and can then deliver the message to the application and issue the ack as soon as it receives all the fragments needed to reconstruct a complete message. This message ack lets intermediate hosts remove from their caches even fragments that the destination has never received.

VI. Comparison to TCP Acknowledgements

As described in Section III, AllNet secure acks offer substantially more assurance of message delivery than TCP acks. Specifically, these secure acks guarantee[4] that the issuer of each ack is either the sender of the data message, which generated the ack in the first place, or a receiver that is in possession of the key needed to decrypt the message. In contrast, TCP acks have no cryptographic or security properties, and can easily be manipulated by an attacker desiring to convince a sender that messages (called **segments** in TCP) have been received even when they have not.

Such assurance is particularly important in an ad-hoc network. Unlike the infrastructure of the Internet, there is little assurance that a node in an ad-hoc network is benign.

This section offers further comparisons between these two types of acks.

TCP acks are counters, referring to the sequence number after the last byte that was received. Because of this, TCP acks can, and in fact do, acknowledge not only the message whose receipt led to the ack being issued, but also all the data that preceded it. TCP ack numbers are 32 bits, so an ack in the original TCP can acknowledge up to 2^{32} different bytes. TCP using the Protection Against Wrapped Sequences [12] (PAWS) mechanism can theoretically acknowledge up to 2^{64} different bytes of data. Note that TCP acks count bytes whereas AllNet secure acks identify packets, so for example a 1,000-byte message requires only one secure ack in AllNet but consumes 1,000 sequence and ack numbers in TCP.

The AllNet secure acks are not counters, so cannot be used as cumulative acks (where one ack potentially acknowledges many, many data packets) as in TCP, but on the other hand a single AllNet message ack can acknowledge many different fragments. There are 2^{128} possible different AllNet message acks. Since these acks are randomly selected, the birthday paradox tells us that the chances of collision increase substantially once the number of acks begins to approach 2^{64}.

In addition, these secure acks might collide with acks from any sender, whereas the TCP connection mechanism limits sequence and ack number collisions to be within a connection.

A. Performance Analysis

Optimistically assuming a 64-bit sequence number space for TCP and with reasonable assumptions against delivery of old packets, TCP is guaranteed not to have sequence or ack collisions as long as 2^{63} or fewer bytes are transmitted on a connection within a two-minute Maximum Segment Lifetime (MSL) period, leading to a maximum bandwidth of over 7×10^{16} bytes/second for each TCP connection.

For the secure acks described in this paper, to stay well away from the birthday paradox we assume that it would be undesirable to have more that about 2^{60} messages alive in the network at any given time. We further assume message sizes of 1,000 bytes and a maximum message lifetime (as specified by the message expiration option in AllNet) of about a week. Satisfying these assumptions limits the entire network to about 10^{15} bytes per second.

On the other hand, for use over the Internet, if we assume a maximum message lifetime of 2 minutes, similar to the MSL of TCP, the network can support almost 10^{19} bytes per second.

While this throughput for the entire network cannot be directly compared to the TCP per-connection throughput, it is clearly adequate for both the current, largely experimental AllNet, and for any foreseeable developments.

Ad-hoc networks are typically small and relatively inefficient [11], and so even in the imaginable future are unlikely to scale to large sizes and large amounts of traffic. Therefore for the ad-hoc side of the AllNet communications, even the lower network throughput derived by assuming a 1-week message lifetime is very abundant.

VII. Future Work and Conclusions

It is clear from the above analysis that if AllNet ever becomes as popular as TCP, it will have to be redesigned or extended for higher performance,

[4]With the usual caveat about guarantees only holding as long as the cryptographic algorithms and the keys and devices are secure.

just as TCP has been and likely will be again in the future.

The secure acks described in this paper provide many advantages over conventional acks such as used in TCP. This paper has explored a few, including the guarantee that the ack can only be issued by a receiver that has the correct cryptographic key, the ability to use the combination of message ack and fragment ack to let the sender know how much of the data the receiver actually needs, and the use of the acks to securely enable removing acknowledged messages from peer caches.

Secure acks as described above only work when sent encrypted. This is not a substantial limitation, since even if the data is sent in the clear, the ack itself may be encrypted. The need for encryption means these secure acks rely on a key infrastructure to identify legitimate receivers entitled to issue each ack.

When encryption is not an option, one could instead imagine having identical ack generators based on identical keys on each pair of sender and receiver, such that the receiver can generate the same sequence of acks as the sender. In such a case, the acks do not need to be transmitted at all. Instead, the sender would include an identifier for the ack associated with a message, and the receiver can independently generate the ack and hash it to compare them any received message IDs. The receiver can then issue a valid ack that can be verified by any peer that is caching messages.

Alternatively, in a scheme somewhat resembling the Bitcoin blockchain [13], and if anonymity is not a concern, a receiver could digitally sign a received message ID with a widely known public key. This scheme requires an infrastructure (perhaps a blockchain?) to record and distribute the public keys, but does not require any encryption. Since a shared blockchain requires a persistent connection to the Internet, this scheme may be not be as suitable for ad-hoc communications as for systems that do not rely on delay-tolerant ad-hoc communications.

In summary, this paper has explored some of the design space for secure and anonymous acknowledgments. While this section has indulged in speculation, the mechanisms in Sections III and IV, and in the initial part of Section V, are fully implemented and live on the AllNet network.

REFERENCES

[1] Hastings and McLean, "TCP/IP spoofing fundamentals", 1996 International Phoenix Conference on Computers and Communications, doi: 10.1109/PCCC.1996.493637.

[2] Kevin Fall, "A Delay-Tolerant Network Architecture for Challenged Internets", SigCOMM, Aug 2003.

[3] Benhamida, Bouabdellah, and Challal, "Using delay tolerant network for the Internet of Things: Opportunities and challenges", 2017 8th International Conference on Information and Communication Systems (ICICS), 2017, doi: 10.1109/IACS.2017.7921980.

[4] Mallorqui, Zaballos, and Serra, "A Delay Tolerant Network for Antarctica", IEEE Communications Magazine, August 2022, doi: 10.1109/MCOM.007.2200147.

[5] Partridge, "The Technical Development of Internet Email", IEEE Annals of the History of Computing, vol. 30, no. 2, April-June 2008, doi: 10.1109/MAHC.2008.32.

[6] Nowitz, "Uucp Implementation Description", Unix Manual Version 7. https://web.archive.org/web/20180221100921/http://a.papnet.eu/UNIX/v7/files//doc/36_uucpimp.pdf

[7] Biagioni, "Ubiquitous Interpersonal Communication over Ad-Hoc Networks and the Internet", 47th Hawaii International Conference on Systems Sciences), in January 2014, and other papers at https://alnt.org/

[8] Biagioni, "Preventing UDP Flooding Amplification Attacks with Weak Authentication", International Conference on Computing, Networking and Communications (ICNC 2019), February 2019, Honolulu, Hawaii.

[9] "Specifications for the Secure Hash Standard", U.S. Federal Information Processing Standards Publication 180-3, October 2008.

[10] Information Sciences Institute, "Internet Protocol", RFC 791 (section 3.2.1.4), September 1981.

[11] Gupta and Kumar, "The Capacity of Wireless Networks", IEEE Transactions on Information Theory, vol. 46, March 2000.

[12] Borman, Braden, Jacobson, and Scheffenegger, "TCP Extensions for High Performance", RFC 7323, September 2014.

[13] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", made public May 24 2009. http://bitcoin.org/bitcoin.pdf