RN/06/05
March 10, 2006

Research
Note

Unforgeable Acknowledgements for Unlinkable Communication

Michael Rogers and Saleem Bhatti

Electronic Mail: m.rogers@cs.ucl.ac.uk
URL: http://www.cs.ucl.ac.uk/staff/M.Rogers/

Abstract

This paper describes a method of acknowledging packets in an untrusted network while maintaining unlinkability. The recipient of a packet generates an *unforgeable acknowledgement* that allows relays as well as the sender to verify that the packet was delivered unmodified to its intended recipient. Unlike digital signature schemes, relays do not need to share any keys with the sender or recipient, or to know their identities or pseudonyms. Acknowledgements enable nodes to measure the level of service provided by their downstream neighbours and optionally to adjust the level of service they provide in return, creating an incentive to forward packets and preventing certain denial of service attacks. Unforgeable acknowledgements are based on standard cryptographic primitives and have low storage, bandwidth and computation requirements for all parties.

Department of Computer Science
University College London
Gower Street
London WC1E 6BT, UK

1 Introduction

Security protocols have traditionally attempted to ensure authentication, accountability and non-repudiation, whereas privacy protocols have attempted to ensure anonymity, unlinkability and deniability. There is an obvious tension between these two sets of goals, despite the similar cryptographic tools used to achieve them. Many security protocols have recently been proposed for peer-to-peer and ad hoc networks, to address the problem of users who consume more resources than they contribute. Encouraging these 'free riders' to cooperate may have a significant impact on the performance and even viability of decentralised networks. Free riding also has security implications, because denial of service attacks often depend on resource exhaustion. Unfortunately, many of the proposed solutions to the free riding problem require detailed record-keeping and information-sharing that could undermine the privacy of users [1, 2, 3]. Other proposals depend on central coordination or identity management, introducing a single point of failure into otherwise decentralised systems [4, 5, 6].

If pairs of nodes can measure the level of service they receive from one another and use this information to adjust the level of service they provide in return, then each node has an incentive to cooperate in order to continue receiving cooperation [7]. This local, reciprocal approach does not require central coordination, record-keeping or information-sharing. Each node must be able to identify and authenticate its neighbours, but identities do not need to have global scope: a node can potentially present a different identity to each neighbour. If the level of service offered to each neighbour is proportional to the level of service received, there is no incentive for a node to present multiple simultaneous identities to the same neighbour [8].

The next section describes a protocol that enables nodes in a packet-forwarding network to measure the level of service provided by their neighbours. By measuring reliability at the packet level, a single incentive mechanism can support a wide range of end-to-end services without relays needing to be aware of the details of higher protocol layers [9]. Our protocol uses end-to-end *unforgeable acknowledgements* that can be verified by relays without establishing a security association with either of the endpoints. Unlike a digital signature scheme, relays do not need to share any keys with the sender or recipient, or to know their identities or pseudonyms, so the protocol supports unlinkable communication. Section 3 demonstrates that the protocol is secure as long as the underlying cryptographic primitives are secure. Section 4 considers possible applications of the protocol, Section 5 reviews related work, and Section 6 concludes the paper.

2 Unforgeable Acknowledgements

The unforgeable acknowledgement protocol uses two kinds of messages: packets, which contain a header and a data payload, and acknowledgements, which only contain a header. The sender and recipient of each packet must share a secret authentication key that is not revealed to any other node, and each packet sent between the same endpoints must contain a unique serial number or nonce to prevent replay attacks. This number need not be visible to intermediate nodes, and indeed the protocol does not reveal any information that can be used to determine whether two packets have the same source or destination, although such information might be revealed by traffic analysis or by other protocol layers.

Unforgeable acknowledgements make use of two standard cryptographic primitives: message authentication codes and collision-resistant hashing. Before transmitting a packet, the sender computes a message authentication code over the packet using a secret key shared with the recipient. Instead of attaching the message authentication code to the packet, the sender hashes the message authentication code and attaches the hash to the packet. Relays store a copy of the hash when they forward the packet. If the packet reaches its destination, the recipient computes a message authentication code over the received packet using the secret key shared with the sender. If the hash of this message authentication code matches the hash attached to the packet, the recipient sends the message authentication code as an acknowledgement, which is forwarded back along the path taken by the packet. Relays can verify that the acknowledgement hashes to the same value that was attached to the packet, but they cannot forge acknowledgements because they lack the secret key to compute the correct message authentication code, and because the hash function is collision-resistant.

More formally, let $H(x)$ denote the hash of x, let $MAC(y, z)$ denote a message authentication code computed over the message z using the key y, and let $\{a, b\}$ denote the concatentation of a and b. Let k be the secret key shared by the sender and recipient, and let d be the data to be sent. The relays between the sender and recipient are denoted $r_1 \ldots r_n$.

The sender first attaches a unique nonce or serial number s to the data, to produce the payload $p_1 = \{s, d\}$. The sender calculates $h_1 = H(MAC(k, p_1))$ and sends $\{h_1, p_1\}$ to relay r_1. Each relay r_i stores the identity of the previous hop under the hash h_i, and forwards $\{h_{i+1}, p_{i+1}\}$ to the next hop, where $h_{i+1} = h_i$ unless r_i modifies the header and

$p_{i+1} = p_i$ unless r_i modifies the payload. On receiving $\{h_{n+1}, p_{n+1}\}$ the recipient calculates $H(MAC(k, p_{n+1}))$ and compares the result to h_{n+1}. If the result does not match, then either $h_{n+1} \neq h_1$ or $p_{n+1} \neq p_1$ – in other words either the header or the payload has been modified by one of the relays, and the recipient does not acknowledge the packet. If the packet has not been modified, the recipient returns the acknowledgement $a_{n+1} = MAC(k, p_{n+1})$ to relay r_n. Each relay r_i calculates $H(a_{i+1})$, and if the result matches a stored hash, forwards a_i to the previous hop stored under the hash, where $a_i = a_{i+1}$ unless r_i modifies the acknowledgement. When a relay receives an acknowledgement that matches a stored hash, it knows that neither the header, the payload, nor the acknowledgement was modified downstream.

It is important to note that while packets may carry source or destination addresses, the acknowledgement protocol does not authenticate these addresses. An unforgeable acknowledgement proves one of two things. To the sender, it proves that the downstream neighbour delivered the packet to its intended destination. To a relay, it proves that the downstream neighbour delivered the packet *to the destination intended by the upstream neighbour* – this does not necessarily correspond to the destination written on the packet. The upstream and downstream neighbours might collude to produce and acknowledge packets with spoofed addresses, so unforgeable acknowledgements cannot be used to discover reliable routes to particular destination addresses. However, in the context of unlinkable communication this limitation becomes a strength: packets need not carry any information to associate them with one another, or with any particular sender or recipient.

There is nothing to stop an attacker from modifying the header of a packet, perhaps replacing it with a hash generated by the attacker for acknowledgement by a downstream accomplice. However, the attacker will then be unable to provide a suitable acknowledgement to its upstream neighbour, and thus from the point of view of the attacker's upstream neighbour the attacker will effectively have dropped the packet and transmitted one of its own instead, albeit one with an identical payload. The upstream neighbour will not consider the attacker to have delivered the packet as requested, and will reduce its level of service accordingly. Likewise if the attacker modifies the payload instead of the header, the recipient will not acknowledge the packet and again the attacker will be unable to provide an acknowledgement to its upstream neighbour. Any modification to a packet or acknowledgement is thus equivalent to dropping the packet, and with respect to reliability measurement a node that modifies packets or acknowledgements is equivalent to a free rider. The protocol does not attempt to distinguish between malicious and accidental packet loss, since attempting to do so might allow dishonest users to manipulate the system.

3 Proof of Security

This section demonstrates that the unforgeable acknowledgement protocol is secure as long as the underlying cryptographic primitives are secure. Four specific assumptions are made about the underlying primitives:

1. It is not feasible to recover the secret key k by observing any sequence of authenticated messages $\{MAC(k, m_1), m_1\} \ldots \{MAC(k, m_n), m_n\}$.
2. It is not feasible to calculate $MAC(k, m)$ for a given message m without knowing the secret key k.
3. It is not feasible to find the preimage x of a given hash $H(x)$.
4. It is not feasible to find a second preimage $y \neq x$ for a given preimage x, such that $H(y) = H(x)$.

The first two properties are standard requirements for MAC functions, and the last two properties (inversion resistance and second preimage resistance) are standard requirements for cryptographic hash functions. We note in passing that these properties are not affected by recent collision attacks on cryptographic hash functions [10, 11].

First we show that the protocol does not reveal the secret key. If an eavesdropper could recover the secret key from some sequence of packets $\{H(MAC(k, m_1)), m_1\} \ldots \{H(MAC(k, m_n)), m_n\}$ and their acknowledgements $MAC(k, m_1) \ldots MAC(k, m_n)$, then the attacker could also recover the key from $\{MAC(k, m_1), m_1\} \ldots \{MAC(k, m_n), m_n\}$, contradicting the first assumption above.

Next we show that an attacker cannot forge acknowledgements without the secret key. Assume that an attacker succeeds in forging an acknowledgement. Either the forged acknowledgement is identical to the genuine acknowledgement, or it is different. If it is identical then either the attacker has succeeded in calculating $MAC(k, m)$ without knowing k, which contradicts the second assumption above, or the attacker has found a way of inverting the hash function, which contradicts the third assumption. On the other hand if the forged acknowledgement is different from the genuine acknowledgement, the attacker has found a second preimage $y \neq x$ such that $H(y) = H(x)$, which contradicts the fourth assumption.

4 Applications

This paper does not describe a complete communication system, but rather a building block that allows nodes to measure reliability without sacrificing unlinkability. The mechanism by which senders and recipients exchange secret keys is not discussed here, because the acknowledgement protocol is independent of the key exchange mechanism; similarly, end-to-end encryption is not discussed, although we would expect it to be used by any parties requiring unlinkability.

Unforgeable acknowledgements can operate in a peer-to-peer overlay or at the network layer, providing an incentive for nodes to forward packets as well as transmitting their own. There are no dependencies between messages other than between a packet and its acknowledgement, so each packet can be treated as an independent datagram; retransmission, sequencing and flow control can be handled by higher protocol layers. This allows a single incentive mechanism to support a wide range of services. In contrast, many existing incentive mechnisms are limited to file-sharing applications, because they require content hashes to be known in advance [12, 13, 14, 15].

We assume that acknowledgements can be forwarded back along the same path as the packets they acknowledge – if the reverse path is not the same as the forward path then acknowledgements can only be verified end-to-end. This could make our protocol unsuitable for use over wireless networks with unidirectional links, for example. We do not assume that links have symmetric bandwidth or latency.

The acknowledgement protocol does not require relays to share keys with senders or recipients, but it can easily be generalised to situations where the sender wishes to direct traffic through a certain trusted relay: the sender exchanges keys with the relay, and the relay exchanges keys with the recipient; the relay acknowledges and forwards the packets it receives from the sender, and the recipient acknowledges the packets it receives from the relay. The recipient does not need to know the identity of the sender, and indeed onion routing [16] could be layered on top of the unforgeable acknowledgement protocol just like any other service.

The bandwidth, storage and computation overheads of our protocol are modest. Each packet must carry the hash of its message authentication code and a unique nonce or serial number, and the sender and recipient must each perform one hash computation in addition to the normal cost of using message authentication codes. Each relay must store one hash per outstanding packet, perform a single hash computation and table lookup per acknowledgement, and forward one message authentication code per acknowledgement. Since acknowledgements are small and there is at most one acknowledgement per packet, acknowledgements can be piggybacked on packets to reduce transmission costs.

5 Related Work

5.1 Reciprocation

Reciprocation between neighbours is used to encourage resource contribution in several deployed peer-to-peer networks [12, 13, 14]. These systems differ in how they allocate resources among cooperative neighbours, but all of them provide a higher level of service to contributors than non-contributors. The level of service received from a neighbour is measured by the amount of data downloaded, and hash trees [17] are used to verify each block of data received.

SLIC [18] is an incentive mechanism for message forwarding in peer-to-peer search overlays. The level of service received from a neighbour is measured by the number of search results it returns, but without a way to verify results this creates an incentive to return a large number of bogus results.

SHARP [19] is a general framework for peer-to-peer resource trading; digitally signed ‘tickets’ are used to reserve and claim resources such as storage, bandwidth and computation. Claims can be delegated, so peers can exchange resources with peers more than one hop away, but the identities of all peers in the delegation chain must be visible in order to validate the claim. This makes SHARP unsuitable for unlinkable communication.

5.2 Authenticated Acknowledgements

IPSec [20] uses message authentication codes for end-to-end authentication at the network layer. This makes it possible to authenticate transport-layer acknowledgements, but the message authentication codes can only verified by the endpoints, not by third parties such as relays.

TLS [21] uses message authentication codes at the transport layer. TCP headers are not authenticated, however, so it is possible for relays to forge TCP acknowledgements; to ensure reliable delivery the endpoints must also

use application-layer acknowledgements. As with IPSec, the message authentication codes used by TLS cannot be verified by relays.

Some robust routing protocols for ad hoc networks use message authentication codes to acknowledge packets and detect faulty links and nodes [22, 23]. This requires a trusted certificate authority for key distribution, and rules out unlinkable communication.

5.3 Authentication using One-Way Functions

Gennaro and Rohatgi [24] describe two methods for authenticating streams using one-way functions. The on-line scheme uses one-time signatures [25, 26]. Each block of the stream contains a public key, and is signed with the private key corresponding to the public key contained in the previous block. The first block carries a conventional asymmetric signature. One-time signatures are large, so the on-line scheme has a considerable bandwidth overhead. The computational cost of verifying a one-time signature is comparable to an asymmetric signature, although signing is more efficient.

The off-line scheme uses chained hashes, where each block contains the hash of the next block, and the first block carries an asymmetric signature. The entire stream must be known to the sender before the first block is sent. This scheme is similar to the use of hash trees in file-sharing networks.

The Guy Fawkes protocol [27] also uses chained hashes. The sender does not need to know the entire stream in advance, but each block must be known before the previous block is sent. Each block A_i carries a preimage X_{i-1} and a hash $H(X_i)$ that are used to verify the previous block, and a hash $H(A_{i+1}, H(X_{i+1}), X_i)$ that commits to the contents of the next block. The first block carries a conventional signature.

Several ad hoc routing protocols use hash chains to reduce the number of asymmetric signature operations [28, 29, 30, 31]. Others use delayed disclosure, in which a hash and its preimage are sent by the same party at different times, requiring loose clock synchronisation [32, 29, 33]. In the present protocol the preimage is not sent until the hash is received, so no clock synchronisation is required.

The schemes described above use similar techniques to the protocol described in this paper, but their aims are different. Whereas the aim of a signature scheme is to associate messages with a sender, the aim of our protocol is to associate an acknowledgement with a packet, without identifying the sender or recipient of the packet. Thus all of the signature schemes mentioned above require an initial asymmetric signature to identify the sender, whereas unforgeable acknowledgements do not require asymmetric cryptography.

6 Conclusion

We have described a protocol for acknowledging packets without revealing the identities of the sender or recipient. The acknowledgements created by the protocol are unforgeable but can be verified by untrusted third parties. The protocol has broad applicability: it can operate at the network layer and does not require relays to establish a security association with the endpoints, or to be aware of the details of higher-layer protocols. It can be seen as a building block for unlinkable communication systems, allowing nodes to measure the level of service received from their neighbours so that they can adjust the level of service they provide in return.

References

[1] T.W. Ngan, D.S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In F. Kaashoek and I. Stoica, editors, *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03), Berkeley, CA, USA, February 2003*, volume 2735 of *Lecture Notes in Computer Science*, pages 149–159. Springer, 2003.

[2] M. Ham and G. Agha. ARA: A robust audit to prevent free-riding in P2P networks. In *5th IEEE International Conference on Peer-to-Peer Computing, Konstanz, Germany*, August-September 2005.

[3] S. Buchegger and J.Y. Le Boudec. A robust reputation system for P2P and mobile ad hoc networks. In *2nd Workshop on Economics of Peer-to-Peer Systems, Cambridge, MA, USA*, June 2004.

[4] L. Anderegg and S. Eidenbenz. Ad hoc VCG: A truthful and cost-efficient routing protocol for mobile ad hoc networks with selfish agents. In *ACM Mobicom*, 2003.

[5] P. Druschel and A. Rowstron. PAST: A large-scale, persistent peer-to-peer storage utility. In *8th Workshop on Hot Topics in Operating Systems, Elmau, Germany*, May 2001.

[6] A. Adya, W.J. Bolosky, M. Castro, G. Cermak, R. Chaiken, J.R. Douceur, J. Howell, J.R. Lorch, M. Theimer, and R.P. Wattenhofer. FARSITE: Federated, available, and reliable storage for an incompletely trusted environment. In *Proceedings of the 5th USENIX Symposium on Operating Systems Design and Implementation, Boston, MA, USA*, pages 1–14, December 2002.

[7] M. Rogers and S. Bhatti. Cooperation in decentralised networks. In *London Communications Symposium, London, UK*, September 2005.

[8] J.R. Douceur. The Sybil attack. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proceedings of the 1st International Workshop on Peer-to-Peer Systems (IPTPS '02), Cambridge, MA, USA, March 2002*, volume 2429 of *Lecture Notes in Computer Science*, pages 251–260. Springer, 2002.

[9] J.H. Saltzer, D.P. Reed, and D.D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–288, November 1984.

[10] X. Wang, D. Feng, X. Lai, and H. Yu. Collisions for hash functions MD4, MD5, HAVAL-128 and RIPEMD, 2004. Cryptology ePrint 2004/199, available from `http://eprint.iacr.org/2004/199.pdf`.

[11] X. Wang, Y.L. Yin, and H. Yu. Finding collisions in the full SHA-1. In *25th Annual International Cryptology Conference, Santa Barbara, CA, USA*, August 2005.

[12] C. Grothoff. An excess-based economic model for resource allocation in peer-to-peer networks. *Wirtschaftsinformatik*, 45(3):285–292, June 2003.

[13] B. Cohen. Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems, Berkeley, CA, USA*, June 2003.

[14] Y. Kulbak and D. Bickson. The eMule protocol specification. Technical report, School of Computer Science and Engineering, Hebrew University of Jerusalem, January 2005.

[15] P. Gauthier, B. Bershad, and S.D. Gribble. Dealing with cheaters in anonymous peer-to-peer networks. Technical Report 04-01-03, University of Washington, January 2004.

[16] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42(2):39–41, February 1999.

[17] R. Merkle. Protocols for public key cryptosystems. In *IEEE Symposium on Security and Privacy, Oakland, CA, USA*, April 1980.

[18] Q. Sun and H. Garcia-Molina. SLIC: A selfish link-based incentive mechanism for unstructured peer-to-peer networks. In *24th International Conference on Distributed Computing Systems*, 2004.

[19] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat. SHARP: An architecture for secure resource peering. In *19th ACM Symposium on Operating Systems Principles, Bolton Landing, NY, USA*, October 2003.

[20] S. Kent and R. Atkinson. RFC 2401: Security architecture for the internet protocol, November 1998.

[21] T. Dierks and C. Allen. RFC 2246: The TLS protocol, January 1999.

[22] B. Awerbuch, D. Holmer, C. Nita-Rotaru, and H. Rubens. An on-demand secure routing protocol resilient to Byzantine failures. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, USA*, pages 21–30, September 2002.

[23] I. Avramopoulos, H. Kobayashi, R. Wang, and A. Krishnamurthy. Highly secure and efficient routing. In *IEEE Infocom, Hong Kong*, March 2004.

[24] R. Gennaro and P. Rohatgi. How to sign digital streams. In B.S.J. Kaliski, editor, *Proceedings of the 17th Annual Cryptology Conference (CRYPTO '97), Santa Barbara, CA, USA, August 1997*, volume 1294 of *Lecture Notes in Computer Science*, pages 180–197. Springer, 1997.

[25] L. Lamport. Constructing digital signatures from a one-way function. Technical Report CSL-98, SRI International, Palo Alto, CA, USA, 1979.

[26] R. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Proceedings of the Conference on the Theory and Applications of Cryptographic Techniques (CRYPTO '87), Santa Barbara, CA, USA, August 1987*, volume 293 of *Lecture Notes in Computer Science*. Springer, 1988.

[27] R.J. Anderson, F. Bergadano, B. Crispo, J.H. Lee, C. Manifavas, and R.M. Needham. A new family of authentication protocols. *Operating Systems Review*, 32(4):9–20, October 1998.

[28] R. Hauser, T. Przygienda, and G. Tsudik. Reducing the cost of security in link-state routing. In *ISOC Symposium on Network and Distributed System Security, San Diego, CA, USA*, February 1997.

[29] S. Cheung. An efficient message authentication scheme for link state routing. In *Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC '97), San Diego, CA, USA*, pages 90–98, December 1997.

[30] M.G. Zapata and N. Asokan. Securing ad hoc routing protocols. In *Proceedings of the ACM Workshop on Wireless Security (WiSe'02), Atlanta, GA, USA*, pages 1–10, September 2002.

[31] Y.C. Hu, D.B. Johnson, and A. Perrig. SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks. In *4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA '02)*, June 2002.

[32] A. Perrig, R. Canneti, J.D. Tygar, and D. Song. The TESLA broadcast authentication protocol. *CryptoBytes*, 5(2):2–13, 2002.

[33] Y.C. Hu, A. Perrig, and D.B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *8th International Conference on Mobile Computing and Networking (MobiCom)*, September 2002.