DSN 2007 - 37th IFIP/IEEE Annual Conf. Dependable Systems and Networks. Edinburgh, UK. Jun 2007.

A lightweight mechanism for dependable communication in untrusted networks

Michael Rogers
Department of Computer Science, UCL, Gower St, London WC1E 6BT, UK
m.rogers@cs.ucl.ac.uk

Saleem Bhatti
School of Computer Science, University of St Andrews, St Andrews, Fife KY16 9SX, UK
saleem@cs.st-andrews.ac.uk

Abstract

We describe a method for enabling dependable forwarding of messages in untrusted networks. Nodes perform only relatively lightweight operations per message, and only the originator and destination need to trust each other. Whereas existing protocols for dependable communication rely on establishing a verifiable identity for every node, our protocol can operate in networks with unknown or varying membership and with no limits on the creation of new identities. Our protocol supports the maintenance of unlinkability: relays cannot tell whether a given originator and destination are communicating. The destination of each message generates an unforgeable acknowledgement (U-ACK) that allows relays and the originator to verify that the message was delivered unmodified to the destination, but relays do not need to share keys with the originator or destination, or to know their identities. Similarly, the endpoints do not need to know the identities of the relays. U-ACKs can be seen as a building block for dependable communication systems; they enable nodes to measure the level of service provided by their neighbours and optionally to adjust the level of service they provide in return, creating an incentive for nodes to forward messages. Our work is ongoing.

1. Introduction

Increasingly, the dependability of a networked communication system is considered a key issue for the operation of a larger system as a whole. However, there are a number of challenges to achieving dependability, including the possibility of malicious behaviour that aims to disrupt or subvert communication. For a set of nodes, N, forming a communication network, we need some way of assessing whether *correct forwarding behaviour* is being observed. Here, our definition of *correct forwarding behaviour* is very simple: forwarding behaviour is deemed to be correct in the network of nodes, N, if a node, $n_i \in N$, the originator, can send a message to another node, $n_j \in N, i \neq j$, the destination, by relying on the message forwarding behaviour of N.

Our scenario is a network of nodes, N, in which we assume that only the originator, n_i, and destination, n_j, of each message trust each other, and there is no other trust relationship within the network. This means that n_i and n_j may not be able to see or verify the identities of any other nodes in the network. Nodes that forward a message but are not the originator or the destination are termed *relays*. In our discussion, we assume that any node may act as an originator of its own messages as well as a destination or relay for the messages of other nodes.

We assume that nodes communicate using the general unit of communication, which we will term a *message*, which is any self-contained block of data. Depending on the application and the layer of operation in the communication stack, a *message* could be a *packet*, a *frame*, a *datagram*, an *application data unit (ADU)* such as a block in a file transfer, etc. Our goal is to enable measurably dependable forwarding of messages in a network of untrusted nodes.

Correct forwarding behaviour can be achieved with high confidence if all the nodes trust each other. Trust may be established, for example, by the use of a certified or certifiable identity for each node. Identities or pseudonyms may also be derived from other information within the network, such as network addresses, but these may be transient or may not be strongly verifiable. In certain circumstances, where identity is available and verifiable to some degree, it may be possible to detect failed, misbehaving or malicious nodes [1, 2, 3].

However, in many environments, it may not be practical to insist on establishing the identity of every member of N to provide the level of trust required to have confidence of correct forwarding behaviour. For example, the

membership of the network may be changing constantly, or it may not be possible to verify or certify the identity of a node, $n \in N$. Even if it is possible to verify a node's identity, that identity may be subverted without being detected. In other cases, users may wish to maintain anonymity or *unlinkability*, meaning that other network nodes should be unable to determine whether a given pair of nodes are communicating. Additionally, maintaining unlinkability helps to counter some denial of service attacks (DoS) which may target nodes based on their identities. Examples of such environments include ad hoc wireless networks, peer-to-peer networks and some online communities.

Another issue especially relevant to ad hoc networks and peer-to-peer systems is resource usage. Many protocols have recently been proposed 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 open membership networks. Free riding also has security implications, because denial of service (DoS) attacks are often based 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 [4, 5, 6]. Other proposals depend on central coordination or identity management, introducing a single point of failure into otherwise decentralised systems [7, 8, 9].

If pairs of adjacent 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 [10]. 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 these identities can be local in scope, and a node is free to 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 [11].

1.1. Structure of this paper

The next section describes the *U-ACK protocol*, which enables nodes in a message-forwarding network to measure the level of service provided by their neighbours. By measuring dependability at the message 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 [12].

Our protocol uses end-to-end (originator to destination) *unforgeable acknowledgements (U-ACKs)* 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 originator or destination, or to know their identities. U-ACKs are a general mechanism designed to be used in conjunction with an *application-specific dependability metric (ASDM)* that is a function of the messages sent and the U-ACKs received.

Section 3 demonstrates that U-ACKs cannot be forged as long as the underlying cryptographic primitives are secure. Section 4 considers possible applications of the protocol, and Section 5 discusses issues that would affect engineering of the protocol. In Section 6 we review related work, and Section 7 concludes the paper and gives a brief description of our ongoing work and thoughts for the future.

2. Unforgeable acknowledgements

The unforgeable acknowledgement (U-ACK) protocol handles two kinds of data: *messages*, which consist of a header and a data payload, and *acknowledgements*. The originator and destination of each message must share a secret key that is not revealed to any other node, and each message 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 messages have the same originator or destination, although such information might be revealed by traffic analysis or by other protocol layers.

Our protocol does not rely upon or mandate any particular key management scheme or key exchange mechanism; any existing scheme appropriate to the application can be used. We only assume that the originator and destination have some way of establishing a shared secret key, k.

2.1. Overview

Unforgeable acknowledgements (U-ACKs) make use of two standard cryptographic primitives: *message authentication codes (MACs)* and *collision-resistant hashing* (or simply *hashing*). Any node can generate a correct hash, but only a node that knows the authentication key can generate a correct MAC. So, before transmitting a message, the originator computes a MAC over the message using the secret key, k, shared with the destination. Instead of attaching the MAC to the message, the originator attaches the *hash of the MAC* to the message. Relays store a copy of the hash when they forward the message. If the message reaches its destination, the destination computes a MAC over the received message using the secret key, k, shared with the originator. If the hash of this MAC matches the hash received with the message, then the destination has validated the message, and sends the *MAC as an acknowledgement*,

which is forwarded back along the path taken by the message. Relays can verify that the acknowledgement hashes to the same value that was attached to the message sent by the originator, but they cannot forge acknowledgements for undelivered messages because they lack the secret key, k, to compute the correct MAC, and because the hash function is collision resistant.

2.2. Description of the protocol

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 concatenation of a and b. Let k be the secret key shared by the originator and destination, and let d be the data to be sent. The relays between the originator and destination are denoted $r_1 \dots r_M$.

The operation of the protocol proceeds as follows:

1. The originator first attaches a unique nonce or serial number, s, to the data, to produce the payload $p_1 = \{s, d\}$.

2. The originator calculates $h_1 = H(MAC(k, p_1))$ and sends $\{h_1, p_1\}$ to relay r_1.

3. Each relay r_m stores an identifier (e.g. the network address) of the previous node under the hash h_m, and forwards $\{h_{m+1}, p_{m+1}\}$ to the next node, where $h_{m+1} = h_m$ unless r_m modifies the header, and $p_{m+1} = p_m$ unless r_m modifies the payload.

4. On receiving $\{h_{M+1}, p_{M+1}\}$ from r_M, the destination calculates $H(MAC(k, p_{M+1}))$ and compares the result to h_{M+1}. If the result does not match, then either $h_{M+1} \neq h_1$ or $p_{M+1} \neq p_1$ – in other words either the header or the payload has been modified by one of the relays – and the destination does not acknowledge the message.

5. If the message has not been modified, the destination returns the acknowledgement $a_{M+1} = MAC(k, p_{M+1})$ to relay r_M.

6. Each relay r_m calculates $H(a_{m+1})$, and if the result matches a stored hash, forwards a_m to the previous node stored under the hash, where $a_m = a_{m+1}$ unless r_m modifies the acknowledgement.

7. When a relay receives an acknowledgement whose hash matches the stored hash of a message it previously forwarded, it knows that neither the header, the payload, nor the acknowledgement was modified by any node between itself and the destination.

8. When the originator receives an acknowledgement whose hash matches the stored hash of a message it previously transmitted, it knows that neither the header, the payload, nor the acknowledgement was modified by any node between itself and the destination, and that the message was correctly delivered to the destination, since only the destination could have generated the acknowledgement.

2.3. Malicious nodes

It is important to note that while messages may carry source or destination addresses, the U-ACK protocol does not authenticate these addresses. A U-ACK proves one of two things. To the originator, it proves that the downstream neighbour delivered the message to its intended destination. To a relay, it proves that the downstream neighbour delivered the message *to the destination intended by the upstream neighbour* – this does not necessarily correspond to the message's destination address, if any. The upstream and downstream neighbours might collude to produce and acknowledge messages with spoofed addresses, so U-ACKs cannot be used to discover reliable routes to particular addresses. However, in the context of unlinkable communication this limitation becomes a strength: messages need not carry any information to associate them with one another, or with any particular originator or destination.

There is nothing to stop an attacker from modifying the header of a message, 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 its neighbour's point of view the attacker will effectively have dropped the message 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 message as requested, and may reduce its level of service accordingly (this will depend on how the application-specific dependability metric is evaluated and used). Likewise if the attacker modifies the payload instead of the header, the destination will not acknowledge the message and again the attacker will be unable to provide an acknowledgement to its upstream neighbour.

With regard to dependability, any modification to a message or acknowledgement is equivalent to dropping the message, and a node that modifies messages or acknowledgements is equivalent to a free rider.

2.4. Lost messages

Messages may be lost, reordered, or modified for a number of reasons, and it may not be possible to determine whether such events are due to the normal behaviour of the

network, or due to the malicious or incorrect behaviour of relays. For example, in a wireless ad hoc network, loss, reordering, bit errors and even duplication of messages may be considered normal behaviour for the network.

In contrast to existing approaches that try to identify the node or link responsible for each failure, we take the pragmatic approach of measuring dependability without attempting to distinguish between malicious, selfish, and accidental failures. This makes it possible for our protocol to operate in networks with a variable failure rate; with an unknown, changing, or open membership; and where the quality of service (QoS) of network parameters is dynamically variable.

3. Unforgeability

The strength and scalability of our system comes from its simplicity. Only originators and destinations can generate a set of check bits for a message, but any node can verify those check bits without needing to know the identity of, or share state with, the originator or destination. The key to our protocol is the unforgeability of acknowledgements, so in this section we demonstrate that relays cannot forge acknowledgements as long as the underlying cryptographic primitives are secure. Four specific properties are listed below with respect to the behaviour of the underlying primitives. These properties are commonly accepted and are based on the design goals of those 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 and design goals for MAC functions, and the last two properties (inversion resistance and second preimage resistance) are standard requirements and design goals for cryptographic hash functions. These properties are not affected by recent collision search attacks on cryptographic hash functions [13, 14]. As long as these properties are true for any specific MAC and hash function used to implement our protocol, we consider U-ACKs to be unforgeable.

First we show that the protocol does not reveal the secret key. If an eavesdropper could recover the secret key from some sequence of messages:

$$\{H(MAC(k, m_1)), m_1\} \ldots \{H(MAC(k, m_n)), m_n\}$$

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 property 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 property above, or the attacker has found a way of inverting the hash function, which contradicts the third property. 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 property.

4. Applicability

This paper does not describe a complete communication system, but rather a protocol building block that allows nodes to measure dependability. The mechanism by which originators and destinations 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 parties requiring privacy and unlinkability. Additionally, an application would need to select an application-specific dependability metric (ASDM) to use with the U-ACK protocol. The ASDM, which will have application-specific semantics, should be a function of the messages originated and/or relayed and the U-ACKs received.

4.1. Generality

Unforgeable acknowledgements can operate in a peer-to-peer overlay or at the network layer, providing an incentive for nodes to forward messages as well as transmitting their own. There are no dependencies between messages other than between a message and its acknowledgement, so each message 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 upper-layer protocols and services. In contrast, many existing incentive mechanisms are limited to file-sharing applications, because they require content hashes to be known in advance [15, 16, 17, 18].

4.2. Reverse path forwarding

We have assumed that the forward path of the message is the same path that will be followed, in reverse, by the U-ACK, i.e. reverse path forwarding is being used. This may not be possible in all networks – for example some wireless networks may contain unidirectional links. Where the assumption of reverse path forwarding does not hold, there are two situations to consider:

- The reverse path has *some relay nodes in common* with the forward path. In this case, there may be some nodes that receive information about the dependability of their neighbours, while others do not, at least not for all messages.
- The reverse path has *no relay nodes in common* with the forward path. In this case, only the originator receives information about the dependability of its neighbours.

In either situation, the U-ACK protocol provides a coarse-grained input to the ASDM: simply that the network as a whole is managing to deliver messages to their intended destinations and that U-ACKs are being returned, i.e. correct forwarding behaviour is being maintained for the network, N. Nodes that act as relays may build up confidence of their neighbours' dependability without having to send messages themselves: whenever a node sees a U-ACK for a message, it knows that the message was successfully delivered, without necessarily having any knowledge of the path beyond its immediate neighbours. This coarse-grained measure of dependability may be sufficient for some applications.

Note that routing asymmetries such as those commonly found in the Internet do not prevent reverse path forwarding: each relay stores the identity of the previous node when forwarding a message, so the reverse path can be found even if the relay's routing tables are asymmetric. Similarly, asymmetric link bandwidth is not a problem as long as it is possible to return one acknowledgement for each message sent in the opposite direction.

Our protocol can therefore operate in situations with diverse routing paths; the ASDM chosen should take account of the nature of the paths and any path information that may be available.

4.3. Gateways, proxies and middleboxes

The U-ACK protocol does not require relays to share keys with originators or destinations, but it can easily be generalised to situations where the originator wishes to direct traffic through a certain trusted gateway, proxy, or other middlebox: the originator exchanges keys with the gateway and the gateway exchanges keys with the destination; the gateway acknowledges messages from the originator and forwards them to the destination with new headers; and the destination acknowledges messages from the gateway. The key shared by the originator and the gateway is independent from the key shared by the gateway and the destination, so it is possible for the gateway to re-encrypt the messages before forwarding them. Indeed, onion routing [19] could be layered on top of our protocol, providing originator anonymity as well as originator-destination unlinkability.

4.4. Non-unicast communication

So far, we have implicitly considered unicast communication. However, there may be further considerations if non-unicast mechanisms are used for message delivery. For example, some protocols in mobile ad hoc networks (MANETs) use flooding-based or broadcast-based forwarding. In such applications, multiple copies of a message may reach a destination or relay node by different paths. To maintain the association between messages and U-ACKs, a simple extension of the protocol is to return a copy of the U-ACK to every neighbour from which a copy of the corresponding message was received. However, this may lead to increased overhead, so an application may wish to reduce the number of U-ACKs transmitted and adjust accordingly the definition and dynamic evaluation of the ASDM being used.

Another issue is that of one-to-many or many-to-many communication, such as network-layer or application-layer multicast. Here, a single transmission may have many destinations, and a naive translation of our protocol would require each of these destinations to send an acknowledgement. Reliable multicast is an area of ongoing research [20, 21, 22], but it is known to be impractical to use per-destination acknowledgements; thus our protocol seems unlikely to be applicable to large-scale reliable multicast without modification.

In a tree-based scheme for multicast distribution, one possibility would be for key nodes in the tree to act as trusted gateways, as described in Section 4.3. Each gateway would be responsible for receiving U-ACKs from nodes below itself in the multicast tree and sending aggregate U-ACKs to a node above itself in the tree. An aggregate U-ACK would indicate delivery of a message to all intended recipients.

However, although solutions for key management and distribution in such scenarios have been defined [23], modifications of this kind would increase the complexity of the protocol, introduce additional overhead, and could lead to a weakening of the overall security and dependability of the system.

Another approach is to look at the way dependability is

handled in other schemes, such as bimodal multicast [21] or QuickSilver [22]. It may be possible to modify the designs of those schemes to incorporate U-ACKs and so permit operation in untrusted environments, but we have not examined this in detail.

We consider the use of U-ACKs in non-unicast communication to be a topic for further study.

5. Engineering considerations

5.1. Timeouts

Relays cannot store hashes indefinitely while waiting for acknowledgements – at some point, old hashes must be discarded to make room for new ones. A relay that receives an acknowledgement after discarding the corresponding hash cannot verify or forward the acknowledgement, so there is no reason for a relay to store a hash for longer than its upstream or downstream neighbours. The most efficient solution would be for all relays along the path to discard the hash at the same time, after waiting an appropriate amount of time to receive a U-ACK. Using a separate synchronisation protocol to determine when to discard hashes is not practical in an untrusted scenario, and adding a time-to-live field to messages would undermine unlinkability by allowing relays to estimate the distance to the originator.

Fixed timeouts avoid these problems while ensuring that adjacent relays discard the hash at approximately the same time, and are simple to implement. The length of the timeout represents a tradeoff between the maximum end-to-end latency the network can tolerate, and the number of outstanding hashes each relay must store. The choice of an appropriate timeout will depend on the application. As an example, TCP's maximum segment lifetime (MSL) represents a conservative estimate of the maximum latency across the Internet: a typical implementation value is 30 seconds, which is much greater than the typical latency or round-trip time, and TCP may wait for a period of two MSLs before allowing re-use of a port number. Thus 60 seconds seems to be a reasonable timeout for hashes in an Internet overlay; shorter timeouts may be appropriate for other applications.

5.2. Overhead

The bandwidth and computation overheads of the U-ACK protocol are modest. Each message must carry the hash of its MAC and a unique nonce or serial number, and the originator and destination must each perform one hash computation in addition to the normal cost of using MACs. Each relay must perform a single hash computation and table lookup per acknowledgement, and forward one MAC per acknowledgement. Since acknowledgements are small and there is at most one acknowledgement per message, acknowledgements could be piggybacked on messages in bidirectional communication to reduce transmission costs.

The originator and each relay must store one hash per outstanding message, so the storage overheads of the protocol depend on three factors: the data rate of the end-to-end path, D_p; the message size, S_m; and the timeout for stored hashes, T_h. If S_h is the size of a hash for a single message, we can approximate the storage requirement of a node, S_n, as:

$$S_n = \frac{D_p.T_h.S_h}{S_m}$$

So, with a 60 second timeout and a minimum message size of 125 bytes including headers, a node with an 11 Mb/s link (e.g. 802.11b wireless LAN) may need to store up to 660,000 outstanding hashes. This would require $\sim$13 MB of memory for a 160-bit hash function such as SHA-1. This represents the worst case, however, when all messages have the minimum size and all acknowledgements take the maximum time to arrive; in a more realistic scenario with a mean message size of 500 bytes and an average round-trip time of 5 seconds, the storage overhead would be just $\sim$275 KB.

A malicious node might attempt to exhaust a relay's memory by flooding it with messages, forcing it to store a large number of hashes. This attack could be mitigated by allocating a separate storage quota to each neighbour; a neighbour that exceeded its quota would then simply cause its own hashes to expire early.

5.3. Measuring dependability

Unforgeable acknowledgements allow nodes to measure the dependability of their neighbours, but the exact way in which the application-specific dependability metric (ASDM) is computed and refreshed will depend on the application; the behaviour of the ASDM in time (including freshness, decay and/or expiry of dependability information) and in space (for a given neighbour, path, or flow) will be application specific, and our protocol places no specific constraints on the nature of this metric. However, to demonstrate that a fine-grained dependability metric does not necessarily require information about the identities of the originator or destination, we offer the following sketch of a flow-based ASDM. This is only intended as an example; other metrics may be appropriate for other applications.

We define a flow as any sequence of messages that have the same originator and destination and that are semantically related in some way – for example, the sequence of messages that make up a single file transfer. To enable flow-based dependability measurement, the originator marks all messages in a flow with an arbitrary *flow identifier*. The contents of the flow identifier are not significant – it is just a label, and it is not covered by the message authentication

code. All messages in a flow are marked with the same flow ID.

Flow IDs have local scope: when a relay forwards a message, the flow ID used on the downstream link may be different from the ID on the upstream link. However, messages belonging to the same flow should still have matching flow IDs on the downstream link. Each flow traversing a link must be assigned a flow ID that distinguishes it from any other flows currently traversing the same link. Flows arriving at a node from different upstream neighbours must be treated as distinct, and must be assigned distinct flow IDs on any downstream link, even if they happen to have matching IDs on their respective upstream links.

The use of flow IDs with local scope is similar to the use of label-swapping in virtual circuits, but there is no requirement to establish flow ID state in the relays before data transfer begins – flow IDs can be assigned to new flows on the fly.

Relays can use flow IDs for fine-grained dependability measurement without needing to know the origins or destinations of the flows. For each flow it is currently forwarding, a relay stores the identifiers (e.g. network addresses) of the upstream and downstream nodes, the flow IDs for the upstream and downstream links, and the application-specific dependability metric for the flow. The ASDM might take the form of a running average of the fraction of messages acknowledged (e.g. an exponential moving average, which can be stored as a single floating-point number). All this information is soft state: it does not need to survive across restarts, and information about inactive flows can be discarded to reclaim space.

6. Related work

6.1. Reciprocation

Reciprocation between neighbours is used to encourage resource contribution in several deployed peer-to-peer networks [15, 16, 17]. 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. Hash trees [24] are calculated in advance and used to verify each block of data received, so these networks are only suitable for distributing static files.

SLIC [25] 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. In contrast, the U-ACK protocol makes it easy to detect bogus acknowledgements.

SHARP [26] 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 trade 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 untrusted environments and unlinkable communication.

6.2. Authenticated acknowledgements

2HARP [2] is a routing protocol for ad hoc wireless networks in which each node that receives a packet sends an acknowledgement to the previous two nodes, allowing each node to verify that its downstream neighbour forwarded the packet. Every node has a public/private key pair for signing acknowledgements; these key pairs must be certified by a central authority to prevent nodes from generating extra key pairs and using them to create bogus acknowledgements. This requirement makes 2HARP unsuitable for use in open membership networks.

IPSec [27] uses message authentication codes for end-to-end authentication at the network layer. This makes it possible to authenticate transport-layer acknowledgements as well as data, but the MACs can only verified by the endpoints, not by third parties such as relays.

TLS [28] uses MACs at the transport layer. TCP headers are not authenticated, however, so it is possible for relays to forge TCP acknowledgements. As with IPSec, the MACs used by TLS cannot be verified by relays.

Some robust routing protocols for ad hoc networks use MACs to acknowledge messages and to detect faulty links and nodes [29, 30]. This requires a trusted certificate authority for key distribution, and rules out unlinkability.

6.3. Authentication using one-way functions

Gennaro and Rohatgi [31] describe two methods for authenticating streams using one-way functions. The first scheme uses one-time signatures [32, 33]. 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 this scheme has a considerable bandwidth overhead. The computational cost of verifying a one-time signature is comparable to that of an asymmetric signature, although signing is more efficient.

The second 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 originator before the first block is sent. This scheme is similar to the use of hash trees in file-sharing networks.

The Guy Fawkes protocol [34] also uses chained hashes. The originator does not need to know the entire stream in advance, but each block must be known before the previous block is sent. Each block carries a preimage and a hash that are used to verify the previous block, and a hash 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 [35, 36, 37, 38]. Others use delayed disclosure, in which a hash and its preimage are sent by the same party at different times, requiring loose clock synchronisation [36, 39, 40]. In our 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 an originator, the aim of our protocol is to associate an acknowledgement with a message, without identifying the originator or destination of the message. The signature schemes mentioned above therefore require an initial asymmetric signature to identify the originator, whereas the U-ACK protocol does not require asymmetric cryptography.

7. Conclusion and future work

We have described the U-ACK protocol, which enables nodes in a network to measure the dependability of their neighbours in forwarding messages using *unforgeable acknowledgements (U-ACKs)*. The protocol does not require trust between all nodes in the network; the only nodes that need to be able to verify one another's identities are the originator and destination. 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 or in a peer-to-peer overlay, 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 dependable communication systems, allowing nodes to measure the level of service received from their neighbours using an *application-specific dependability metric (ASDM)* that is a function of the messages sent and the U-ACKs received.

We are currently investigating specific properties of the protocol when used in peer-to-peer systems, e.g. the dynamics of resource usage that occur with a mixture of free riders, altruists and reciprocators. The investigations will explore the sensitivity of the U-ACK scheme to various parameters such as the size and structure of the network, the proportion of free riders, etc.

The U-ACK scheme could also have applicability to systems that need to be robust to Byzantine failures, such as applications for safety-critical systems, civil defence and military use.

References

[1] R. Perlman. Network layer protocols with Byzantine robustness. PhD Thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, August 1988.

[2] P.W. Yau and C.J. Mitchell. 2HARP: A secure routing protocol to detect failed and selfish nodes in mobile ad hoc networks. In *Proc. 5th World Wireless Congress, San Francisco, CA, USA*, pages 1–6, 2004.

[3] D. Quercia, M. Lad, S. Hailes, L. Capra and S. Bhatti. STRUDEL: Supporting trust in the dynamic establishment of peering coalitions. In *Proc. 21st Annual ACM Symposium on Applied Computing (SAC2006), Bourgogne University, Dijon, France*, 23-27 April 2006.

[4] T.W. Ngan, D.S. Wallach, and P. Druschel. Enforcing fair sharing of peer-to-peer resources. In F. Kaashoek and I. Stoica, editors, *Proc. 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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 *Proc. 5th USENIX Symposium on Operating Systems Design and Implementation, Boston, MA, USA*, pages 1–14, December 2002.

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

[11] J.R. Douceur. The Sybil attack. In P. Druschel, F. Kaashoek, and A. Rowstron, editors, *Proc. 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.

[12] 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.

[13] 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`.

[14] 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.

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

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

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

[18] 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.

[19] 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.

[20] The IETF Reliable Multicast Transport (RMT) Working Group. `http://www.ietf.org/html.charters/rmt-charter.html`

[21] K. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky. Bimodal multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, May 1999.

[22] K. Ostrowski, K. Birman, and A. Phanishayee. QuickSilver scalable multicast. Technical Report TR2006-2063, Cornell University, April 2006.

[23] S. Rafaeli and D. Hutchison. A survey of key management for secure group communication. ACM Computing Surveys, 35(3):309–329, September 2003.

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

[25] 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.

[26] 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.

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

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

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

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

[31] R. Gennaro and P. Rohatgi. How to sign digital streams. In B.S.J. Kaliski, editor, *Proc. 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.

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

[33] R. Merkle. A digital signature based on a conventional encryption function. In C. Pomerance, editor, *Proc. 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.

[34] 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.

[35] 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.

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

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

[38] 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.

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

[40] 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.