Who goes here ?

Confidentiality of location through anonymity

Ian W. Jackson
ijackson@chiark.greenend.org.uk
Cambridge University Computer Laboratory / ORL, Cambridge

April 1997
revised February 1998
Special Edition

Declarations

I declare that this dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration.

I also declare that this dissertation is not substantially the same as any that I have submitted for a degree or diploma or other qualification at any other University, and that no part of this dissertation has already been or is being concurrently submitted for any such degree, diploma or other qualification.

This dissertation does not exceed 60,000 words in length.

Contents

List of Figures

Chapter 1. Overview

Initially, a specific problem in the security of location is presented: that of allowing the subjects of a particular location-tracking system to retain control over the distribution of the information about their location.

Prior work in the field of location hiding and traffic flow confidentiality is reviewed and discussed.

The social constraints which make the problem interesting and relevant are described, and the trust models and goals of the parties involved in the application area are discussed. The particular set of trust models and goals of interest for the research presented is stated.

The general nature of the problem, and the key difficulties in solving it, are identified. Several possible solutions are outlined. One of these solutions is chosen — the use of mix-based remailer networks, with some modifications — and the choice is justified with reference to the factors operating in the application area.

The cryptographic mechanisms required to implement this solution are discussed, and protocols are proposed which satisfy the application's basic requirements for users' control over their own location tracking information. More sophisticated requirements are discussed and some examples of mechanisms that could be used to implement them are presented, including an 'anonymous certificate of location'.

An attempt at formal analysis of these protocols is made; existing formal methods are shown to be inadequate, and a new logic for the analysis of protocols is presented. The new logic is shown to be useful in general, but still not to be able to express fully the concepts required to analyse the protocols proposed here.

The solution and protocols chosen are developed into a design for a practical system in the initial application area. The design has been partially implemented and tested. Measurements of the performance of this implementation of the practical system are presented and discussed.

The requirements for the mechanism by which users specify the security policies relating to information about their location are presented. A simple but general mechanism and language are proposed; some of their limitations are discussed.

Possible weaknesses in and attacks against the system are described and evaluated, and the most likely failure modes identified.

The conclusions of the work are presented and future research directions suggested; other application areas which might benefit from similar technology are indicated.

Chapter 2. Problem area and inspiration

This will describe the location-tracking system which inspired this work, point out a key weakness in it, and identify the nature of the general problem underlying that weakness. Later we shall see how this problem can be solved.

This work was originally inspired by the Active Badge system [ORL92] developed by ORL in Cambridge (then Olivetti Research Limited and now the Olivetti and Oracle Research Laboratory).

In this system users each wear a small badge, which contains a microprocessor and one end of an infra-red communications link. The badge communicates with transceivers on the walls of the offices and corridors, and the data from the transceivers is collected centrally.

With this system users can easily be found, by asking the active badge database where they were last seen. The staff at ORL generally find this system invaluable. Face-to-face conversations in person can be very effective, and the Active Badge system allows staff to find each other quickly and easily when they want to talk to each other; it also provides good hints as to whether a person is already busy by (for example) showing how many other people are there with them. The system has found many other uses, and the laboratory has come to rely on it.

However, there is little control over who can retrieve the information about where another person is, how often they can do this, and what uses they can put the information to. This leads to fear on the part of badge wearers that the information will be abused, for example by their boss to dock their pay for time spent away from their desk, by burglars to find when to rob their houses, by unwanted visitors to track them down and even perhaps by their spouses to discover their affairs.

2.1. The Olivetti Active Badge System

The badge is a small rectangle, about 5cm on each side. It has two buttons which are used only for some more advanced features. It has a light sensor which is used to disable the badge's transmissions, saving its battery, when it is dark (for example, face down, inside the wearer's pocket, or in a desk drawer), and a small IR transceiver pair. There is, of course, a clip to attach the badge to the wearer's clothing, though some people prefer to wear theirs on a chain around their neck.

Each wall-mounted transceiver is connected via an RS232 serial cable to one of a set of machines which act as managers for the RS232 network and as protocol relays/converters. In times past these were dedicated 68000-based VME rack systems with hardware similar to that for a terminal server; more recently the RS232 links from the transceivers have been connected to the serial ports of dedicated small PC-based UNIX systems.

The current front-end to the badge system is based on the ANSA distributed computing framework; a ANSA agent connects to the relay/converter systems and each relay/converter forwards all messages from the badges to this agent. The agent stores the information in a database and makes it available on request to its clients.

Several client applications are provided, including X windows and text-mode browsers giving a list of all the badges and their locations as well as command-line programs for making a one-off query about a particular badge's location. Facilities are available for setting an 'alert' on a badge so that the application will beep and pop up a window (or make some other form of notification) when that particular badge is seen to change its location.

The front-end ANSA framework does not make the information available other than within the 'same' or 'related' organisations, as defined by the system's administrators. However, there is little or no access control at many stages of the system; the back-end data collection network in particular has no built-in access control mechanism. It is likely that a malicious person could gain information about the locations of badges by (for example) connecting to the back-end relay/converter directly; the author has been able to verify that at least raw copies of messages transmitted by badges, with the identity numbers for the badges and transceiver locations, are available to anyone on the global Internet with a small amount of technical knowledge about the protocols, addresses and port numbers in use.

There are no explicit mechanisms for a badge wearer to limit the distribution of information about the whereabouts of their badge other than the per-organisation restriction mentioned earlier, which is weak and ineffective even on its own terms. There are no mechanisms available for badge wearers to determine how often or to whom the information about their location is being made available.

Gateways to 'finger' and to the World Wide Web have been set up which make the information available to third-party callers in a nicely-presented fashion; these interfaces do log the identity of callers, and badge wearers are able to configure (for example) that their badge should beep when someone makes such an enquiry about them. However, the information available to them about the identity of the enquirer is scant, due to the general lack of mandatory strong identification on the Internet, and there are limited facilities for conditionally denying access on a per-user or per-enquirer basis.

In another deployed Active Badge system at the Cambridge University Computer Laboratory, the World Wide Web and finger gateways do not answer requests from users with network addresses outside the University. However, these policy restrictions are sometimes bypassed by third-party gateways within the University which may pass the information outside — such gateways may well be of a very general nature (for example, a universal finger-by-Web gateway), and their administrators not be aware of all the data being passed through their gateway or even of the fact that there are some University organisations with a security policy violated by the existence of their system.

The badges can provide other useful services to wearers, which make them reluctant to abandon them. For example, arrangements have been made for the doors to the building to unlock automatically when a badge is detected nearby, so that wearers do not have to use keys regularly when passing between floors of the building through a public stairwell (though carrying them is still advisable in case of system failure); a wearer can use the facility to have a badge play a short tune to inform them[1] of the arrival of new email and even of its nature; events of vital importance such as meetings, afternoon tea or Tuesday morning sticky buns are announced via the playing of tunes on badges; non-wearers might miss these.

Nevertheless, some badge holders choose not to wear their badges; some have been known to use them (for example) as a flag to indicate whether they are in their office or elsewhere, by turning them face-down when they leave their desk.

This solves the privacy problem for that badge holder, but does not solve the problem for their colleagues who wish to find them. The lack of control by badge wearers over the information gathered makes it hard for them to implement a good compromise between privacy and availability — if they are unhappy with the wide distribution of their location data they are forced to the crude recourse of preventing the information from being gathered at all.

2.2. Location Tracking and Access Control

In order to solve this problem the user of any location-tracking device must be able to control the dissemination of the information about their location. Empirically users and potential users have widely varying views on this subject, and to satisfy them it must be possible to accommodate all reasonable policies. Furthermore, the user must be justifiably confident that the information collected is not available through a 'back door' which bypasses the access control mechanism — a subject about which users frequently express not entirely ill-founded scepticism.

In any system where the user can specify the policy controlling the distribution of information about their own location this policy will have to be stored and implemented by one or more systems trusted by the user, on which relevant programs will speak for the user and grant or deny access to the information.

Ideally the user would only have to specify the policy once, on a single system, even in a system where location tracking has become pervasive and there are multiple competing and cooperating networks of fixed tracking equipment.

Somehow, the location of the badge has to be communicated to these entities but not to any untrusted party or system. This general problem is closely related to traffic flow confidentiality and confidentiality of identity.

The user's location is established for the benefit of the legitimate holders of this information by communications between their mobile computing device(s) and the transceiver

[1] I follow well-established English practice in using the terms 'they', 'their' and 'them' as gender-neutral singulars, as well as plurals [OED89, vol.XVII, p.928, col.3, 'they', 2nd sense] [Churchyard97].

networks and supporting infrastructure. Without the user having a trusted presence at every location they may visit it may be hard to avoid having to reveal the fact that there is *someone* at the user's location, and no-one at other places where there is indeed no-one.

However, as we shall see, if there are many users and the identity of the badge can be concealed then unauthorised observers can be rendered unable to determine the location of any *particular* user, and even if there is only one user in a particular location for a little while the observers cannot necessarily determine whether or not it is continuously the same user.

2.3. Summary

An existing system has been described and some problems in it identified; this thesis aims to present solutions to them. In the next chapter prior work in the field will be reviewed; thereafter the problem will be analysed more formally to find a solution.

Chapter 3. Existing approaches to anonymity and traffic hiding

Several schemes have been proposed before to hide the senders or recipients of messages, providing untraceability or anonymity. These schemes vary according to the nature of the application. This chapter will discuss this prior work and its relationship to my thesis.

The work which will be presented later is parallel in nature to some of the previous work; however, due to differing circumstances the prior work is not directly applicable.

3.1. Classification of requirements

A few attempts have been made to classify the kinds of requirements found in identity hiding systems.

In [SMA95] Didier Samfat, Refik Molva and N. Asokan suppose the main parties to be the mobile user, their home domain which manages their location-independent identity, and a remote domain which manages the network infrastructure near their location. Additionally, there may be some parties with legitimate dealings with the user who may be entitled to see certain information that should be protected from the eavesdroppers (the attacker).

They classify identity hiding requirements according to a matrix whose entries specify which of these parties should be able to identify the mobile user and the home and remote domains. They present a sequence of five matrixes giving different sets of answers to these questions, with increasing levels of untraceability.

According to their classification scheme our requirements are to protect all information about the mobile user from everyone except the home authority (the location of the user's policy, and thus the legitimate owner of the information about the user's location)[1].

In [PW87] Andreas Pfitzmann and Michael Waidner have attempted to classify anonymous or pseudonymous addressing, in terms of whether the address of the anonymised entity is kept secret or published, and according to whether it explicitly states the entity's (public) identity and/or location, and if it does not, whether even the address itself is visible to an attacker.

---

[1]This is their matrix C_5 : $\begin{pmatrix} & H & R & L & E \\ f & 0 & 0 & 0 & 0 \\ h & 1 & 0 & 0 & 0 \\ r & 0 & 1 & 1 & 1 \end{pmatrix}$

where f, h and r are the pieces of information which might be observed – the user's full identity, home affiliation and remote domain, respectively – and H, R, L and E are the entities which might know the information – the home domain, remote domain, legitimate network entities and eavesdroppers, respectively. 1 indicates visibility.

The anonymous address labels for the user's agent which will be presented later (Section 6.1, page 24) are, according to this classification, implicit (meaning that they do not state the identity visibly), private (not linkable to the holder), and therefore temporary (otherwise an attacker could link subsequent messages with the same label and derive information they should not have).

These classification schemes do not identify the key differences between the environments of the various anonymous communication schemes — the differences which mandate the use of new protocols for new applications. These important differences depend mainly on the type of traffic to be carried, the network bandwidth available between various parts of the system, and similar factors.

3.2. Previously proposed schemes

This section will describe a number of the identity-hiding communication schemes that have been proposed in the literature.

3.2.1. Remailers (also known as mixes) and related relaying schemes

In [Chaum81] David Chaum proposes a scheme for hiding the senders of messages, which prevents the identity of the real sender of a particular message from being determined.

In its simplest form, Chaum's scheme has a single trusted entity, which he calls a 'mix', to which all the senders send their messages, encrypted with the mix's public key. Each message contains the identity of the intended recipient and the actual message data. Periodically, the mix decrypts all its incoming messages, sorts them lexicographically in order of the now-visible data and recipient, and sends the resulting messages out in a single batch.

Chaum enhances his scheme by proposing the use of several mixes. Each message passes through a series of mixes, each of which processes the message together with a number of others in the same batch. This makes the scheme secure against collusion by any incomplete subset of the mixes.

Chaum's scheme requires that mixes keep a log of messages they have received, to avoid them retransmitting outputs when inputs are replayed — otherwise, an attacker could replay inputs and observe which outputs are repeated, thus learning the connection between the incoming and outgoing forms of a message at a particular mix.

This scheme has proven itself to be a good basis for (sometimes extensive) adaptation to many different application areas:

In [PW87] an application of mixes to ISDN is described. ISDN users requires high reliability and availability, and a reasonably high bandwidth channel with low delay; furthermore, ISDN has a large number of endpoints. Once a channel (an ISDN connection) has been established, the bandwidth available to its user must remain constant. This leads Pfitzmann and Waidner to propose a scheme where subnetworks with untraceable addressing connect to each other via a traditional traceable backbone. The routing of a connection through some number of mixes is done once, at the time of establishment of an ISDN connection; from then on all communications along that channel are encrypted using stream ciphers and follow the same path, unlike most other mix-based schemes. The paper also contains discussion of how to achieve fault-tolerant mix networks, and a discussion of how to use the dining cryptographers technique (see section 3.2.2 below) to achieve the required synchronous operation.

In [mobip96] a proposal is described for providing location anonymity to mobile IP hosts. They describe a straightforward mix-based scheme, and determine by measurement that its delay is too large for their application. Instead, as with the ISDN mix scheme referred to above, longer-term connection state is established at each location visited so that after the initial connection setup using public-key cryptography subsequent messages are sent through the same path and use only symmetric-key algorithms.

In [onion96] and [onion97] Paul Syverson, David Goldschlag and Michael Reed present an essentially mix-based scheme for hiding the origin and/or destination of TCP/IP connections. Once again, connection routing is established using public-key cryptography at setup time, and remains fixed for the duration of a connection while symmetric key algorithms are used to transmit the data from then on.

In [babel96] Ceki Gülcü and Gene Tsudik describe an application of mixes to Internet electronic mail. There is a description of anonymous return addresses, and some analysis of the vulnerabilities in and attacks possible on such systems.

[SMA95] contains a description of a remote authentication mechanism which essentially uses previously-visited remote domains in a way very similar to mixes. Authentication is chained from one domain to another; the mechanism does not support communication between the mobile user and their home base — rather, it allows the mobile user to take their authentication with them without requiring them to be subject to tracking by anyone but a collusion of domains.

3.2.2. Dining cryptographers — superposed sending

In [Chaum88] David Chaum presents a system (the ‘Dining Cryptographers protocol’) which allows participants to transmit messages to the other participants and yet remain unconditionally untraceable. The scheme involves each participant sending their own message and arranges for the recipients to receive the bitwise sum of the transmitted messages; key material shared between various subsets of the participants is used to blind the origin of the messages. The scheme provides a shared channel, and arbitration mechanisms to prevent excessive collisions are presented. Chaum also discusses ways in which disruptive participants (those who transmit even though they have not negotiated the use of those portions of the shared channel using the arbitration mechanism) can be identified.

David Chaum's scheme depends on the availability of a perfectly secure non-anonymous broadcast, where a recipient of a broadcast can know that every other participant received the same version and from whom the broadcast originated. This is not a realistic assumption for many real broadcast facilities.

In [Waidner89] Michael Waidner extends this work, providing a number of variants on a fail-stop reliable broadcast mechanism (constructed using only unicast messages), integrated into the superposed transmission scheme.

3.3. Attacks on various proposed schemes

A number of weaknesses in some of these variants have already been published:

In [PP89] Birgit and Andreas Pfitzmann show how the use of mixes directly with RSA, as was initially proposed in [Chaum81], is vulnerable to an active chosen-ciphertext attack on RSA. They present a number of possible solutions; oddly, they fail to present the most obvious solution — to use a hybrid cryptosystem where a per-message symmetric block or stream cipher key is generated and used to encipher the actual message, and only this message key is encrypted using the RSA public key algorithm. This defeats the attack by avoiding any interesting interactions with special properties of RSA.

There are a number of more mundane attacks on mix-based systems; Lance Cottrell describes several in [Loki95]. Various factors, including message size and arrival time, can sometimes be correlated by an attacker. Correlation based on arrival time can be hindered by delaying messages in a pool and sending a random output message every once in a while; correlation based on size can be hindered by ensuring that all messages are of a fixed size. Lance Cottrell presents an arrangement for ensuring this — by careful use of constant-size blocks and repeated superencryption he is able to ensure that messages do not shrink as they pass through a remailer.

[Loki95] does not mention the possibility that an attacker might correlate message frequency at endpoints. He does not advocate the sorting of messages into lexicographical order, despite the fact that this is mentioned in [Chaum81].

3.4. Summary

A fair amount of work has been done already — much of it based on David Chaum's mix networks [Chaum81] — on providing sender and recipient untraceability. The nature of the protocols used depends strongly on the exact application area; certain applications require low-delay high-bandwidth connections and do not attempt to hide the association between subsequent messages in a connection. Other applications require point-to-point one-off unlinkable messages, or do not even require the transmission of messages at all, only the authentication of a mobile user.

As we shall see in the next chapters, the application area which is the subject of this thesis has its own set of practical requirements, which drive the decisions about the design of the protocols and system to support them.

Chapter 4. Trust model, parties, security policies

This chapter will identify the parties in the location tracking situation described in the last chapter, and describe the trust relationships between them and the equipment involved. The requirements for a solution, in terms of this trust model, will be specified.

Throughout, 'trusted' will be used to refer to equipment, software and people upon whom the enforcement of a security policy has been deliberately allowed to depend. Any computer security system has components whose compromise defeats the objectives of the system; it is important to identify these components explicitly, and to try to reduce the set of trusted components to the smallest set of most trustworthy components possible.

When we say that someone 'trusts' something we mean either that the component is trusted, as above, or that the person considers the component trustworthy — unlikely to fail through malice, error, or attack. When it is not clear from context which of these is meant, it will be stated.

In general, the aim is to ensure that the information about the location of a person is not disclosed against their will. Thus, the person whose trust is required is the subject of the location tracking technology.

4.1. The mistrust gap; introduction to the parties

The Active Badge system is available from Olivetti as a product, and its deployment is often considered by other instutions. However, it often faces strong resistance from people who feel threatened by the idea that their employers, supervisors, spouses, sexual and social partners, governments, might become able able to 'spy' on them. People feel that detailed logs of their location over a period of time might well reveal any embarrassing secrets they might have, and find the idea objectionable in principle as well; they are also worried that their management might (for example) try to make their pay or promotions dependent on how much time they spend at their desk, or similar absurdities.

This problem of mistrust does not tend to occur so much in small, friendly organisations where the information about location tracking will be available only to other members of the organisation, as it does in large ones with formal and very hierarchical management structures. It is especially acute when pervasive global location tracking is considered.

There are three main parties to be considered, with different objectives:

- The person whose location is being tracked will be called the **user**. Their objective is to distribute the information about their location to people according to whatever the user's own personal policy is; the user may have entered into agreements (with other people and organisations) to be trackable and/or available at certain times or when in certain places, in which case these considerations will form part of their policy (insofar as the user wishes to abide by them).

- The person or organisation who wishes to know where a user is, or to be informed when they arrive at a certain place, or to have access to other information about the user's location, will be called the **enquirer**. Their objective is to obtain the information in question. Sometimes they will be happy to trust information provided by the user; at other times they will want a non-repudiable statement from the user or a statement from a trusted third party, often the infrastructure provider.

- The person(s) or organisation(s) providing the transceiver network and supporting infrastructure will be called the **operator**. In some cases the operator might be an arm of the enquirer's organisation (for example, an employer might run a tracking infrastructure to locate its employees and visitors); on the other hand, perhaps many organisations will cooperate to run commercial public networks allowing users to publish and enquire about their and other people's locations on a commercial basis.

4.2. Persuading reluctant users

The resistance from users described above is often a problem for the deployment and success of active badge and similar location tracking systems.

In some organisations (many military ones, for example) the management are able to decide these matters by fiat, and can mandate that users be subject to a particular location tracking technology. However it is often the case that the organisation needs to persuade the potential users that their privacy is not being invaded. In these organisations the users are otherwise likely to be uncooperative; if the management attempt to impose a solution the users may resort to 'civil disobedience' such as not wearing badges even if issued, or deliberately rendering them inoperative, or even taking legal action or industrial action.

In the case of the pervasive interlinked public and private location tracking networks we might see in the future, the network operators will need to persuade their potential users that the information about their location will be properly protected; otherwise, they will have few customers.

The scheme which will be described will only be useful to operators who are acting in good faith, and have only legitimate reasons for wanting to deploy location tracking schemes (for example, within an organisation to allow that organisation to function more efficiently because colleagues find it easier to meet and communicate, or in the context of a public network to provide the information only to the legitimate users and not leak it as a 'value-added service to priority customers'), and who wish to deploy the scheme with the willing cooperation of the users.

For any scheme such as will be described to be useful it must at least be possible in principle to convince the users that the scheme will do what it claims and that the fact of (for example) wearing badges is not in itself unacceptable. If the users are dead-set against any location tracking at all then all the good will and technical solutions in the world will not help.

If the operator's true intent really is to spy on the users they will of course still face the problem of convincing the users that they are not doing this. The scheme which will be presented later is no help here: users might well be convinced that the scheme does not allow the operators and unauthorised enquirers to spy on them without their consent or knowledge; however, the operators would not be able to satisfy their objectives using this scheme, because the users' conviction mentioned above would be justified. Such an operator would have to look elsewhere — this case will not be discussed further here. In any case, it would be unethical to support such duplicity.

4.3. Trust model — not trusting the network

In order to convince these sceptical potential users it will be necessary to adopt a trust model which assumes at least opportunistic hostile intent by the network infrastructure operator(s). If location tracking becomes widespread there will be multiple competing operators, many of whom the user will not even know of, let alone trust; the user may visit foreign countries whose governments they distrust, or rival companies, or simply have a general distrust of the large commercially-driven telecoms organisations which are likely to become the operators of location-tracking networks.

Even with a simple intra-organisation tracking scheme the network is under direct or indirect control of the organisation's management, and so should not be trusted. After all, if the management and other users could be relied on to keep their promises — even if they could be sure that failure to do so would not lead to discovery — then the users might be happy with a simple assurance that the information would not be misused. Otherwise, if the scheme can easily and undetectably be subverted by applying pressure to just a few system administrators then it is not trustworthy.

This does not mean to say that the network operators are hostile; simply that the scheme needs to guard against the situation where the network, or part of it, *is* .

It is not necessary for it to be impossible for the network operators and other providers of infrastructure to be able to subvert the security; however, it should be difficult for them to do so without running a substantial risk of being detected. If they are detected then users must be able to show that they were attempting to subvert the security, so that it is clear that they have violated the formal or informal agreement between management and users which form the basis upon which the system was deployed.

Users should be able to choose their own security level as well as their own security policy. Since the tracking infrastructure is shared between all users its security must be high, or it must not be trusted, to avoid imposing a low level of security on all users.

The users must be justifiably convinced that the system does not offer 'back doors' for unauthorised persons. The best way of achieving this is to put the trusted portions of the system under their control.

4.4. Operation amongst peers (no distinct users/enquirers/operators)

The users, enquirers and network operators need not necessarily be distinct; the members of a peer-relationship-based organisation may wish to be able to find each other, but still each individually retain control over how information about their location is distributed, and jointly maintain the infrastructure network which supports the tracking. Even if members of the group are generally trustworthy, misuse of the data by a single person could prove very damaging.

The scheme to be described may also be appropriate in this situation. In this case there is no question of the 'operators' or 'enquirers' convincing the 'users' to accept the system; instead the members of the organisation need to be provided with a system that meets their needs both as users desiring privacy and as enquirers.

4.5. Trusted equipment

It follows from the discussion above, and from general good practice in computer security, that the amount of computing and other equipment that is to be trusted should be kept as small as possible.

In this particular environment the software and equipment which is trusted should be subject to the control, administration or at least inspection of the user or someone they find trustworthy. This means that the user (or their delegate) should preferably be able to inspect the software and equipment, and/or have financial, legal or administrative control over it. Even in the case of a small hardware unit with embedded software the user should be in a position which would make it hard for anyone but them to install new software and facilities (and to remove them again to avoid detection if a user becomes suspicious).

This rules out the use of (for example) active badge fixed transceivers and the data acquisition network connecting them to the rest of the world as trusted components of the system. These are easily modified and/or reprogrammed by systems and network support staff, giving an easy line of attack through 'social engineering'.

This requirement also makes the use of shared central server computers problematic, depending on how much the users trust the shared systems' administrators and how tight the security is on those systems. A system with a high degree of openness is both good and bad here: it is good because it is hard for the administrators to subvert it without being detected, but it is bad because it may not provide much protection against the use of operating system security deficiencies to bypass altogether the security provided by the location tracking confidentiality scheme.

In the active badge system the badges are constantly carried by the users, have little capacity for software and have very limited I/O capabilities. Furthermore, they are at the 'most remote' end of all the equipment involved in location tracking. These properties would apply also in some measure to a user-held smartcard.

Therefore, in the scheme described here, the mobile device carried by the user is one piece of trusted equipment. In any case, any secure communication system requires trusted hardware and software at both ends of the communication, and it would therefore be extremely difficult, if not impossible, to avoid requiring the user to carry a piece of trusted hardware.

In some systems the mobile device would be supplied by the network operator. However, if location tracking becomes widespread and many public networks become available, this would ideally be purchased by the user as a commodity item from a general retail outlet, much as a mobile telephone is at the moment. Public specification of all the relevant protocol details would both allow interoperation between different networks, extending the usefulness of the system, and provide mechanisms by which users could audit and/or replace the mobile component.

In any system where the user is able to specify their own access control policy for the information about their location, it will be necessary for the user to have some fixed (as opposed to mobile) trusted entity, where the user's policy resides and is interpreted and implemented. This entity (or entities) will probably have to have knowledge of the user's real location, so that they can answer queries about it. To keep the trusted part of the system as small as possible it makes sense to have no more than one trusted fixed entity.

Again, this entity needs to be as much under the oversight of the user as possible, either legally/organisationally or physically, so that they have control over its operation and the ability to verify it.

Public tracking network operators and other third parties might offer to provide the relevant infrastructure as a commercial service; however, it is important that the user be able to choose the provider of this service so that they can choose one that they trust or even make their own arrangements.

An employee with a friendly employer, for example, might prefer to use a general-purpose workstation on their desk to provide the fixed trusted endpoint in a globally pervasive tracking system. Such a user will often have at least some administrative control over it, and because of their physical proximity they may be able to observe unusual activity. Exactly how good a solution this is will depend on the relationship between the employee and their employer, organisational policies about the administration of computer equipment, and (for example) whether the user's workstation is likely to be be accessed and/or attacked by someone other than the user.

Another alternative is a dedicated piece of trusted hardware, such as a small computer, shared between and run by or on behalf of a number of users. Such a system might reside anywhere with a suitable network connection; often, again, the user's employer may be able and willing to host such a system to make it easier for the user to take up location tracking technology. If this solution is adopted to avoid trust problems related to the organisation's computer system administration policy, then it will be necessary for this trusted computer to be exempted from the parts of the policy that are problematic — typically those which mandate that all computer systems, or all of a particular type, are to be administered by some group of central staff.

Users with permanent network connections to their homes, as may become common in the medium term, should of course be able to use any computing systems in their homes for this purpose.

As we shall see later, due to the technology used by any particular scheme for hiding the user's location it may also be necessary for each user to trust, to some degree, at least a number of other systems.

Each of these other systems may not in itself be able to compromise the user's security. As in multiple-key systems, key-sharing schemes [Shamir79] [Simmons88] or byzantine agreement systems [LSP82] [RB94] security can be achieved by joint action of several independent parties, so that collusion between two or more of them is required to violate the security of the system and gain unauthorised access to information about the user's location.

This collusion can be made less likely by careful choice of the administrators of these other systems; for example, administrators with strong mutual distrust will be less likely to be willing to make themselves vulnerable to each other by cooperating to violate the user's security.

4.6. Query handling and access control

It is also important, when trying to convince users that the system will meet their requirements and is generally acceptable, that the users feel that the system places them in a position to observe and control what information about them is being distributed. If they do not have this feeling the users will become unsure as to whether the system is delivering on its promises.

The best way to provide this feeling is for the user to be able to control and observe directly the entity which distributes information about them (their 'agent'). The agent, since it is a trusted component, should run on the trusted fixed system described in section 4.5.

If the system ensures that only the user's agent can obtain for itself the information about the user's location then the user can control its distribution directly and obviously. In this case all queries for the location of a particular user must be referred to the agent for that user. This will, of course, enable the user to cause their agent to lie about their location. If the enquirer finds this unacceptable then more complicated protocols can meet their needs; this is discussed later, in section 6.9 (page 33).

The agent needs either to be able to aquire the information (the user's location) to answer enquiries, or to have the information itself. If it does not have the information itself it will need to talk to the mobile device to find it out, and this means it needs to know where the mobile device is so that it can properly address its messages.

Therefore the user's agent needs to continuously track the location of the mobile device (and hence the user), so that it can answer the queries itself. The next chapter will demonstrate how this can be achieved.

Chapter 5. Required communications and how to support them

This chapter discusses the nature of the problem, in terms of what kind of communication and information-hiding is required. Possible ways of achieving this information-hiding are listed, and one is chosen to form the basis of a solution. Its key practical problem (the reliance on public-key cryptography) is discussed; later, we shall see how it can be overcome.

The mobile device, the only component of the system which travels with the user, will have to be closely involved in tracking the user. Any mechanism for determining where the user is will require communication by or with the mobile device.

If this communication were plainly sent from the particular mobile device and routed through a network to the appropriate agent then the user could easily be tracked by traffic flow analysis: the user is at the location corresponding to the current physical address of the mobile device. If the mobile portion of the communications link is a broadcast system this locates the user to within the broadcast domain (this may be a wireless LAN subnet, or a mobile telephone cell, for example).

So, something more sophisticated is required to hide the flow of data from mobile device to agent. Furthermore, more advanced applications such as notifications of email or other events of interest to the user will require the mobile device to receive messages from or via the agent.

Broadly speaking there are four possible solutions to hiding traffic flows:

5.1. Cover traffic

It is sometimes possible to hide traffic flows simply by providing cover traffic between many of the different endpoints in the system. If the cover traffic cannot be distinguished from the real traffic then it is hard to find out who is really communicating with whom.

However, cover traffic does not satisfy the requirements of this application, because the user's mobile device is in a particular location and so can only generate cover traffic there. If the user were willing to trust the transceivers that actually communicate with their mobile device, then they could have those generate the cover traffic, but trusting these transceivers would involve trusting physical and organisational infrastructure of which they are a part.

In environments where the user is willing to trust the transceiver network then my scheme is not appropriate. If this is the case then more straightforward ways of providing the required confidentiality are available. For example, the mobile device could merely announce its identity, and the transceiver network could route the message internally and deliver it to the user's agent.

On the other hand if, as described above, we do not wish to trust the transceiver network and its operator, another solution is required.

Cover traffic can also suffer from scaling problems. Every transmitter in the network (in this case, every mobile device) would have to send data to every possible receiver, or at least to a large proportion of the possible receivers, so as to hide which receiver they are really communicating with.

Thus cover traffic on its own is not a solution to the problem; however, as we will see later, it will form part of the solution.

5.2. Dining cryptographers — superposed sending

There are schemes which allow parties to communicate with one another without others in the system being able to observe who is the sender (or recipient) of the messages. These involve simultaneous superposed transmission using cryptographic techniques [Chaum88], and were previously discussed in section 3.2.2 (page 8)

These schemes are not suitable for the current application because they require that endpoints send and receive much more traffic than that for which they are the final sender or recipient. Small mobile devices typically have only very small amounts of communications bandwidth available.

Furthermore, these schemes are in a sense a sophisticated form of cover traffic (a form providing unconditional security) and suffer from many of the same scaling problems.

5.3. Physical broadcast

If a physical broadcast medium is available, where every receiver in the system can be guaranteed to receive every message, but where no receiver can identify the sender of a message, there is a simple solution.

Note that the broadcast discussed here is different to that required by Chaum's dining cryptographers scheme. Chaum's scheme requires a non-anonymous broadcast and constructs from it an anonymous broadcast scheme that we might wish to use; here we consider the availability of a physically secure anonymous broadcast mechanism, and the possibility of using such a mechanism directly.

The mobile device could obtain its location by querying anonymously a local piece of infrastructure, using a conventional non-broadcast transmission technique, and then broadcast this information encrypted in such a way that the user's access control agent is able to decrypt it.

Each agent must carry out a trial decryption of each message it receives, and will be able to read the ones intended for it.

This solution suffers from a scaling problem. Every message sent on the broadcast medium must be received by all the endpoints on the broadcast medium and each endpoint must carry out a trial decryption. In a large system this would quickly outstrip the bandwidth limitations of plausible broadcast media, and also the computational power of a small mobile unit.

It is essential that the destination of each message is not readily available to the other endpoints; it is not a problem if the source is available through triangulation or similar physical techniques, because an observer cannot determine which user each transmission relates to.

There are some other problems with this approach: it may be possible to find the locations of users without going through proper channels, by selectively denying service to receivers in certain physical areas (using jamming, for example) or by using nonstandard aerials and tracking devices to locate transmitters of particular messages.

A series of overlapping broadcast areas (such as 'cells') is not sufficient, as this allows tracking of which cell a user is in, thus revealing useful (but not complete) information about their actual location. There must be one unified broadcast domain.

Additionally, with current technology a radio transceiver capable of broadcasting over fairly large areas is heavy and bulky and has large power requirements. This makes it unsuitable for use in applications like active badges, where the user is expected to carry it around with them or even clip it onto their clothes. Additionally, there is currently some (admittedly not well-substantiated) public concern that the use of moderately powerful radio transmitters very close to people may be dangerous; this is a particular problem in this application as the mobile device may be very close to the user and transmitting for a large proportion of the time.

The engineering problems with mobile broadcast devices can be overcome by having the mobile device merely transmit an encrypted packet (which must be different each time) using conventional means to the local sensor, which would attach the relevant location and use its own broadcast transmitter to send the combination of the two. The agent would be able to decrypt the appropriate packet and record the attached location.

Unfortunately this solution still has the scaling problem of requiring every agent to decrypt every mobile device's messages. This becomes particularly bad if location tracking becomes a universally-used service.

In order to solve the problem in the absence of a true anonymous physical broadcast we must find a way of approximating it, or at least provide the blinding associated with this broadcast, using only a reasonable number of point-to-point communications.

5.4. Remailers

David Chaum describes a scheme where messages are repeatedly encrypted and sent from one relay to another to hide their origin and destination [Chaum81]; this was previously discussed in section 3.2.1 (page 7).

In this scheme, incoming messages to remailers ('mixes') are encrypted using a well-known public key for the remailer. The remailer decrypts the message, and finds the network address of a further remailer and an inner message encrypted using the public key of that next remailer. The message is 'unpeeled' as it is sent on, and eventually the final destination finds the actual message rather than the address of another remailer.

Messages are constructed at the sender by repeated encapsulation. Remailers queue messages into batches which are sent out in a predetermined order, and padding is inserted, so that related message pairs going into and out of a remailer cannot be linked. The system is secure, if enough traffic uses it, even if one or more of the remailers are compromised. So long as the path taken by a message includes at least one secure remailer the security is preserved. Cover traffic is still also required between remailers and to and from endpoints in order to frustrate more sophisticated traffic flow analysis.

At first sight this scheme is very promising; the mobile device could query the local transceiver infrastructure to determine its location and then construct a nested message using this technique with which to inform its agent of its location.

In related schemes a long-term anonymous address can be constructed which contains not only a next-hop address and further encrypted routing information, but also a public key with which to encrypt a payload. These labels can be used to send a payload to the 'owner' of the address without knowing its identity; each remailer encrypts the payload another time as it unwraps a layer of the address, so that the final recipient receives the payload multiply-encrypted. (This encryption of the payload prevents association of incoming and outgoing messages by looking for identical payloads.)

This variant is even more useful, as we can avoid the preliminary exchange of messages between the mobile device and the local transceiver. The mobile device can directly transmit an address label to which the transceiver can append the location as a payload.

Thus, each message sent by the mobile device contains a nested address indication, which at each stage indicates a remailer to which the message should be forwarded and information encrypted using that remailer's public key.

In a way, this is similar to a broadcast, except that the future reception of the message at any endpoint is probabilistic rather than certain, from the point of view of non-omniscient observers; although the scheme serves to deliver the message to the correct endpoint an outside observer cannot identify which message it is so that it appears as if the message might be delivered anywhere.

The repeated transmission of a single long-term anonymous address label would quickly erode its supposed anonymity. The anonymous addressing schemes described above were intended to provide long-term pseudonyms, and do not prevent the identities of the recipients of subsequent messages sent to the same anonymous address from being linked.

However, a user cannot help having their physical location becoming known occasionally. This should not compromise their security. Therefore it is a requirement that individual communications from the same mobile device cannot be linked by an outside observer. This can be achieved by using many different short-term labels.

Remailers rely on delaying messages and sending them out in batches to avoid an attacker being able to use timing to correlate incoming and outgoing traffic. This batching can be a problem for remailers in real-time applications. The batching period must be such that each batch contains many messages with diverse paths; with a sufficient level of diverse traffic it can be made sufficiently small, as we can see in the performance measurements presented later.

The batching provides security by hiding the relationship between incoming and outgoing messages in a batch; it therefore relies on a sufficient number of messages (not controlled by an attacker) existing in each batch. This will probably require the generation of cover traffic to ensure that each batch has enough traffic that is opaque to an attacker so that the attacker cannot determine the relationship between incoming and outgoing messages.

Thus, of the four techniques described — cover traffic on its own, dining cryptographers, physical broadcast and remailer networks — remailer networks (supported by cover traffic) are the best avenue for a solution.

5.5. Restrictions on mobile computation and communication

As it stands the remailer-based scheme indicated above is not suitable because of its extensive use of public-key cryptography.

Public key schemes such as RSA generate large messages, typically many hundreds of bits long, and several nested layers of encryption would produce messages with lengths best measured in kilobytes.

However, communication with very small mobile entities is expensive. For example, current very low power infra-red transmission technology can achieve only around 9600 bps at the physical layer (less actual data can be transmitted since error control and framing are also required, and this bandwidth is shared by all the transmitters in a room). Even more problematic is the power consumption. Transmitting large quantities of data consumes much battery power.

It is not acceptable to have to change the battery in something like an active badge more than a few times a year, and being able to do so less frequently would be a great advantage. This restricts transmission to no more than about a hundred bytes every minute or so; more can be sent but only if it is sent less frequently.

Because of the unreliable nature of the transmission medium many messages will get lost. A retry interval of more than a minute or two would reduce the ability of the system to determine where someone is now, rather than ten minutes ago.

Therefore our messages must be small, and public key confidentiality algorithms like RSA must be avoided for messages transmitted by the mobile device.

Also, the mobile device may well have insufficient computational power for extensive use of public key cryptoalgorithms; though modern mobile devices capable of carrying out RSA signatures and encryptions are now being made small enough to carry around easily, their power consumption requirements when performing several public key encryptions per minute are likely to be difficult to deal with.

For this reason too, public key cryptography is to be avoided.

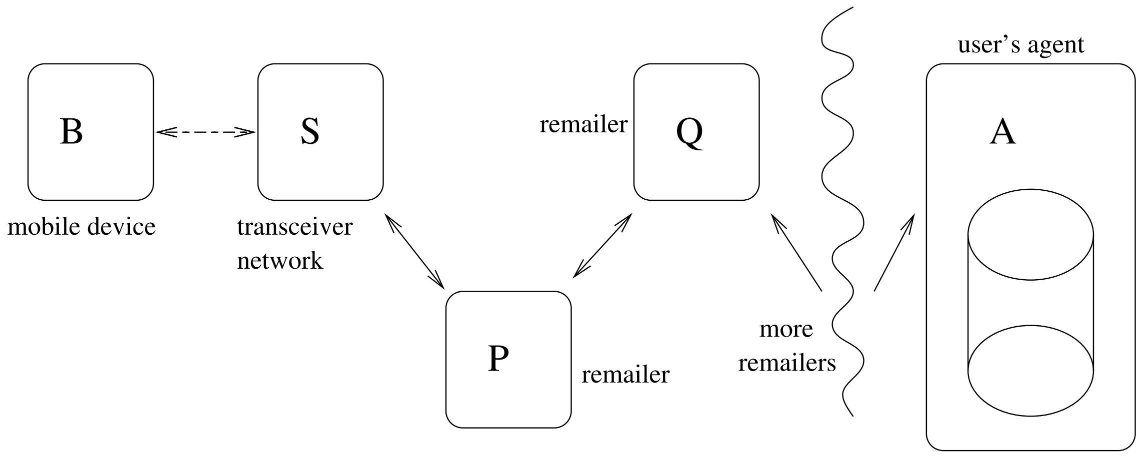
However, as we shall see in Section 6.3 creative use of symmetric key cryptography can obviate the need to use public-key cryptography. First, however, a public-key based based protocol will be presented.

5.6. Summary

The use of a remailer (mix) network can be used to provide the required traffic flow confidentiality; the other approaches considered are not appropriate in this situation.

Ordinary public-key cryptography based remailer schemes use too much communication bandwidth (and perhaps processing power) to be used directly, due to the large messages generated by public-key encryption. However, we shall see that symmetric key cryptography can be made to serve.

Figure 5.1. Overall structure



Chapter 6. Cryptographic protocol design

This chapter will describe cryptographic protocols which implement a solution to the problem at hand. First, a broad description is given, then a concrete protocol using public key cryptography. Public key cryptography being undesirable, a way to achieve the same effect without it is presented. Replay and timeliness in this protocol are discussed. More advanced application requirements and ways of supporting them, including a few more concrete protocols, are presented. A few practical limitations of the scheme are discussed.

Please see Appendix B (pp.87) for expanded examples of these protocols.

6.1. Broad overview — anonymous address labels

Messages are sent from the mobile device to the agent via a transceiver (which handles the mobile communication link) and a series of remailers [Chaum81], all of which must be compromised for a message to be traceable.

The mobile device sends to the transceiver an ‘address label’, which the transceiver can use to communicate with the user’s agent without knowing where that agent is or whose it is. The transceiver knows its own location, and includes this in the body of the message it sends.

The address label consists of a key with which to encrypt the location data, the name (or network address) of the next remailer, and an piece of encrypted instruction which is to be interpreted by that next remailer. This latter will usually consist of another address label and more nested data.

The sending mobile device constructs the address label by repeated encapsulation, and transmits it to nearby transceiver(s). The transceiver(s) which receive it use the label to send a message containing the location of the sighting.

Each remailer strips off, uses and discards one layer of addressing encapsulation. At each stage the key from that stage’s address label is used to re-encipher the data, so that the destination receives a multi-layered encrypted message.

Address labels have to be constructed so that many different labels can be found for a single path or destination — this is so that the address label doesn’t become an alternative means of identifying the user. Preferably, each transmission from a mobile device will use a different label.

6.2. Public key done by mobile device

First, a version of the protocol where public key cryptography is used wherever it seems appropriate will be described. This will help us to understand what it is that we are trying to achieve, by providing an 'ideal version' of the protocols which we might use if the hardware constraints on mobile computation and communication were not so strict. Having established the intent of the exercise, this public-key based protocol will then be refined into one using only symmetric-key cryptography.

Each remailer receives a message in two parts. The second part is the message itself (possibly already encrypted, perhaps more than once).

The first part is encrypted, using the remailer's public key. It contains a symmetric key with which to encrypt the data, and information which can be added to the message to allow the final recipient to determine that key (the key must change with each new label, so that it cannot be used to trace the user and is not so easily compromised). The first part also contains the name of the next hop destination, as well as an encrypted lump that the next hop (presumably) knows how to unpick.

I propose to generate the 'hop session key', which is used to re-encrypt the data, by encrypting a nonce under a key shared between the mobile device and the agent. By encrypting the nonce in two different ways we can give the remailer the session key as well as allow the remailer to pass on (without understanding) information that will allow the agent to recover the session key.

Let us call the mobile device B and the user's agent A. Supposing that remailer P receives:

$$\boxed{\{\{N_{B,P}\}_{K_{AB}}, \{f\,(N_{B,P})\}_{K_{AB}}, Q\,,\ \{E_Q\}_{K_Q}\}_{K_P}}\,,\ \boxed{D}$$

Here the left hand box is the "anonymous address label", and the right hand box is the data to be sent. f might be some simple function such as exclusive-or with a fixed bit-pattern.

E_Q is the information which Q will use to determine what to do with the message; if Q is not the final recipient then E_Q will be an anonymous address label in the form of the left hand box above, containing keys to be used for the data, the identity of the next recipient R, and $\{E_R\}_{K_R}$. Thus the anonymous address label format is recursive, with one layer of recursion being stripped off at each stage.

P then sends to Q (which is either the next remailer or A, the final destination, but P doesn't need to know which):

$$\boxed{\{E_Q\}_{K_Q}}, \boxed{\{N_{B,P}\}_{K_{AB}}, \{D\}_{\{f(N_{B,P})\}_{K_{AB}}}}$$

A can now recover the original D by decrypting $N_{B,P}$, applying f, and hence deriving the key $\{f\,(N_{B,P})\}_{K_{AB}}$.

Before a message as shown above arrives at the first remailer it is assembled by the transceiver. The transceiver (S) receives from the mobile device data equivalent to that used by a remailer (i.e., E_S, which includes amongst other things P and $\{E_P\}_{K_P}$). This data need not be encrypted — we do not trust the transceiver and can better spend effort encrypting things to be decrypted by a more trusted entity. S encrypts the location L using the rules for D above, and sends the result to P.

At the last stage, where $Q = A$, E_Q is merely B's name, so that A knows which of the possibly many mobile devices it is handling is involved. A strips off the many levels of encryption to recover the original D, i.e. L, the location.

An expanded example can be found in B.1, starting on page 87.

6.3. No public key in mobile device

For the reasons described earlier, in section 5.5 (page 22), the use of public key cryptography in the mobile device, and the transmission of public-key encrypted or signed messages to or from the mobile device, is to be avoided.

The required aspect of public key cryptography is the fact that many senders can encrypt messages to the same recipient in such a way that none of the senders can read each other's messages, nor can they relate different messages from the same sender. We can achieve much the same effect by using symmetric key cryptography with large numbers of different keys.

The recipient issues {indicator,key} pairs on request; these are constructed so that the recipient can derive the key from the indicator easily, but the indicators are unpredictable, and no-one else can see the relationship between indicators and keys that they haven't explicitly been sent:

$$I_{Xi} = \{\,i\}_{K_{XX}};\, K_{Xi} = \{f(i)\}_{K_{XX}}$$

We can now rewrite the remailing function using these pairs. For the moment we shall assume that the mobile device has been supplied with suitably constructed anonymous address labels made up with these pairs; how it acquires these will be described later.

Supposing that remailer P receives:

$$\boxed{I_{Pi}\,, \{ I_{Ak}\,, K_{Ak}\,, Q\,, I_{Qj}\,, \{ E_Q \}_{K_{Qj}} \}_{K_{Pi}}}\,, \boxed{D}$$

It then sends to Q:

$$\boxed{I_{Qj}\,, \{ E_Q \}_{K_{Qj}}}\,, \boxed{I_{Ak}\,, \{ D \}_{K_{Ak}}}$$

An expanded example can be found in B.2, starting on page 88.

6.4. Delivery of address labels to mobile device — reverse path

The above scheme requires that the mobile device have a stock of anonymous address labels for its user's agent. Since the labels now only contain symmetric keys and symmetric key encrypted data they are a lot smaller — just over 3 times the key size plus the size of an address, per hop — and the mobile device is now no longer required to manipulate them. This means we can do away with the complication surrounding $N_{B,P}$ and K_{AB} in the version using asymmetric cryptography.

Instead, we have the agent collect indicator-key pairs from the remailers, construct address labels for itself, and send them to the mobile device. It needs a way of sending the labels to the mobile device, anonymously.

This could be achieved by using the remailing facility described above to construct a path in the other direction, towards the mobile device. However, this would mean many round trips for the messages as indicator-key pairs are collected from remailers, first from those nearby in the chain which is being set up, and then, via those, from others.

Instead, we use public key cryptography here. This will require the remailers and agents to have adequate computational resources and communications bandwidth, but this is not a problem for fixed systems.

Supposing that remailer Q receives

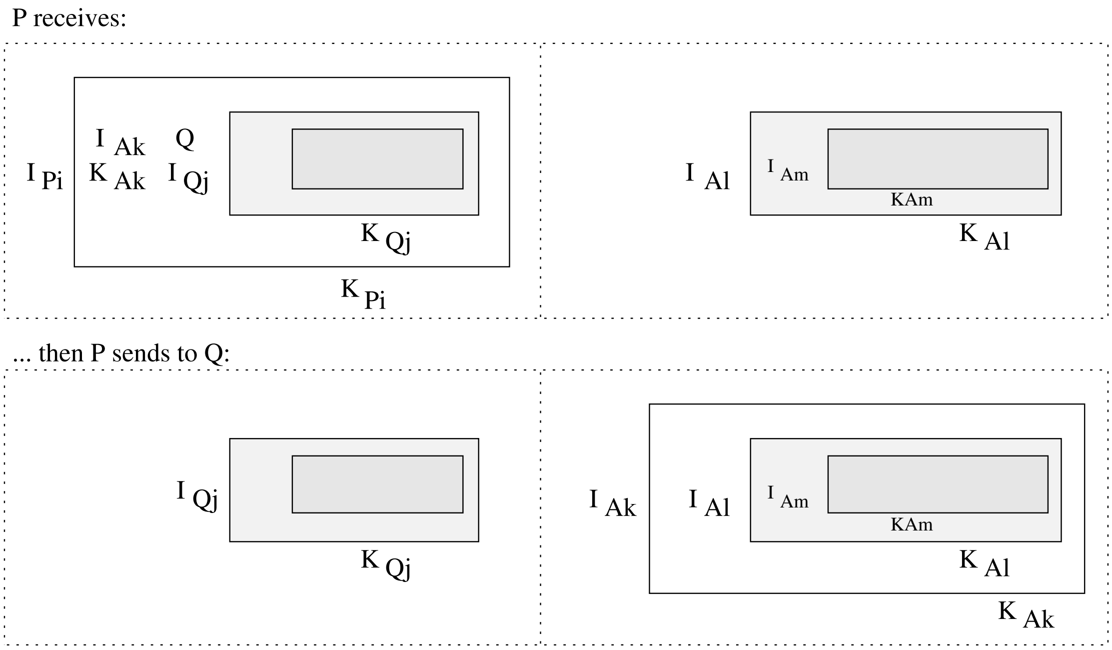
$$\{ P, \{ E_P \}_{K_P} \}_{K_Q}$$

it sends to P:

$$\{ E_P \}_{K_P}$$

This scheme is similar to more conventional remailers – the sender (here the user's agent) is responsible for constructing the message and nesting the encryptions.

Figure 6.1. One remailing step



The final remailer sends the message to the transceiver where the user's agent believes the mobile device is, and the transceiver sends the message on to the mobile device. The innermost message is encrypted using a straightforward symmetric key shared betweeen the mobile device and the agent.

6.5. Kernels of the label and data, and interaction with the transceivers

Transmission of labels to the mobile device is accomplished by the agent sending to the transceiver infrastructure controller, via nested public key remailings, a set of labels. Each label comes with an associated nonce, which is used by the mobile device when it replies.

So A would send to the transceiver infrastructure controller S (via remailers, using the reverse path public-key based mechanism in the preceding section):

$$\{ L, \{ N_{Al}, label_l, N_{A(l+1)}, label_{(l+1)}, \ldots, N_{A(l+n)}, label_{(l+n)} \}_{K_{AB}} \}_{K_S}$$

where L is the location where the message is to be delivered, and K_S is the transceiver's public key. The transceiver will arrange for the $\{\ldots\}_{K_{AB}}$ portion to be broadcast at the appropriate location. Then the mobile device B uses its copy of K_{AB} to decrypt the contents, and adds each $N_{Ax}, label_x$ pair to its stock (where x ranges from l, the first label in this batch, to $l + n$, the last).

The nonces N_{Ax} are used by A to ensure that the data portions of the returning messages are bound to the address portions; each label has the corresponding N_{Ax} buried within it as described below; it will eventually be recovered by *A*.

The address label is constructed by the user's agent by repeated encapsulation of a message destined for itself. In the innermost layer is E_A:

$$\boxed{N_A , B , T_A}$$

where *B* is the identity of the mobile device and T_A is a timestamp generated by the agent when the label is constructed.

When a transmission from the mobile device finally reaches the agent the label will have been unwrapped and the agent can recover E_A. The nonce N_A which is contained in each label, and also accompanies the label in the message to the mobile device delivering the labels, links the two portions of the message. Only messages whose two copies of N_A match, i.e. whose data and label portion were sent together by the mobile device, will be accepted.

As discussed in the section above on the use of public key cryptography, the mobile device initially sends a 'bare' address label to the transceiver, since the transceiver is not trusted any more than the broadcast medium and encryption would be redundant. The initial message is:

$$\boxed{I_{Ak} , K_{Ak} , Q , I_{Qj} , \{ E_Q \}_{K_{Qj}}} , \boxed{\{ N_A , x \}_{K_{AB}}}$$

Here *x* is the number of additional fresh anonymous address labels the mobile device would need to be fully stocked up. This can be used by the agent to decide whether to reply to the mobile device with some more labels.

The transceiver controller will use the address label (the first portion) in the usual way. The data portion is treated specially: the transceiver controller adds its own timestamp and the location. To *Q* is sent:

$$\boxed{I_{Qj} , \{ E_Q \}_{K_{Qj}}} , \boxed{I_{Ak} , \{ T_S , L, \{ N_A , m \}_{K_{AB}} \}_{K_{Ak}}}$$

6.6. Timeliness and acceptance of location data

When the transmission from the mobile device finally reaches the agent the agent recovers from E_A the T_A inserted when the label was generated. It uses this to determine whether the message is an old message being replayed (this is conventionally called 'freshness'; however, here I use that word to mean that an anonymous address label has not been used previously — such labels must be reused, when the mobile device runs out).

The agent will only consider the location in the message as overriding any previous location it has stored for the purposes of supplying to enquirers if T_A is not too out of date by its own clock. A certain amount of slop for stale data must be allowed, as the mobile device may use labels out of order, or they may be reordered in transit.

If it accepts a location it will publish the transceiver's time from the data portion, T_S, as the time at which the user was spotted at that location — unless T_S is earlier than the creation time of the label or later than the agent's idea of the current time, in which case the label creation time or the current time will be recorded instead (this is to cope with any clock skew between the agent and transceiver controller, and as a sanity check).

The agent will always reply to (but not necessarily believe for any other purpose) a claimed statement of location by the mobile device, by sending some labels for the mobile device to that location, as described above — even if the claim it has received uses very old labels and timestamps. This is so that if the mobile device runs out of fresh address labels it is able to use old ones to bootstrap itself. This does not compromise the security, because an observer who retransmits an old label will not know which agent is responding, except by traffic flow analysis; this traffic flow analysis is prevented by transmitting cover traffic between remailers and to and from agents, and by limiting the frequency with which reused labels are acted upon.

The mobile device keeps a stack-like structure containing labels. Newly-received labels are pushed onto the top of this stack. When a label is used it is taken from the top of the stack and reinserted at the bottom. This ensures that the fresh labels are used first, but that failing that the labels which have been used the least recently are used. When the mobile device has enough labels and receives some more, so that the stack becomes overfull when the new labels are added at the top, the most-recently-used labels are pushed off the bottom of the stack and discarded.

The interval between each successive pair of transmissions by the mobile device is randomly chosen from an exponential distribution. This prevents an observer from linking transmissions from the same device by knowing the retransmission interval — the transmissions arrive independently with a mean density of $\frac{1}{\lambda}$ where λ is the mean of the retransmission interval distribution. An observer can determine an expected figure for the number of mobile devices present by observing the transmission density, but cannot link individual transmissions.

6.7. Replay as an attack on the anonymity

There is a further potential vulnerability: if a remailer is re-presented with an address label it has already processed it will process it again in the same way, generating a very similar outbound message. In particular, the inner address label uncovered at the remailer is identical, and can be seen to be so by an attacker.

Therefore, if an attacker can cause messages to be replayed early in the chain of remailers through which a message passes they will be able to observe the repeated messages later in the chain.

However, this information will only be meaningful if they are able to correlate the repeated messages at the end of the chain with those at the beginning. This will be true if the variation which the attacker can cause in the number of times a message passes through the end of the chain is larger than the random variation in the number of times any message is likely to arrive there.

In order to prevent this condition from being satisfied, we must provide random variation in the number of times a message is processed, and limit the effect an attacker can have on this number.

We can do the former by having remailers randomly resend messages with a certain probability. The remailer must act as if the same incoming message had been seen again, and encrypt the data portion once again rather than storing and using a previously encrypted copy, so that the data portions are not linkable.

We can do the latter by having remailers discard messages when they are repeated too much. A remailer must keep track of the labels it has seen, and when it receives the same label again it must throw it away if it has been seen recently.

If we wish absolute security against replay attacks we must make the retransmission of a label by a remailer independent of the number of times it receives it. The remailer should keep the label for a certain length of time, retransmitting it periodically. If a new piece of data arrives attached to the same label it should encrypt that data as the label specifies and transmit it instead of the old data at the next retransmission. After a number of retransmissions (determined randomly, or in advance, but in any case independently of the number of times the label is received) the remailer discards the label in favour of a checksum which it stores and uses to discards any future repetitions.

6.8. Request for indicator-key pairs

The user's agent needs indicator-key pairs in order to construct anonymous address labels. This has to be done anonymously too, so that a remailer handling an address label does not know what its ultimate destination is. We use the 'reverse' anonymity mechanism to transmit the request, and the anonymous address label scheme itself to return the reply.

When a remailer P receives:

$$\{ \boxed{P,\ z,\ N_A} , \boxed{I_{Ak}, K_{Ak}, I_{Qj}, Q, \{E_Q\}_{K_{Qj}}} \}_{K_P}$$

it replies, via Q, with y indicator-key pairs (with y no more than z, the number requested):

$$\boxed{I_{Qj}, \{E_Q\}_{K_{Qj}}}, \boxed{I_{Ak}, \{y, N_A, I_{P(l+1)}, K_{P(l+1)}, \ldots, I_{P(l+n)}, K_{P(l+y)}\}_{K_{Ak}}}$$

Once again the nonce N_A is used to bind together the two parts of the message. The innermost layer of the address label contains as E_A:

$$z,\ P,\ T_A, N_A$$

The agent can use this to check that it has been issued with the correct number of indicator-key pairs and that they are more recent than its request, and also to identify which remailer sent the indicator-key pairs — only the remailer requested will have access to the nonce N_A so that none other can issue labels which will be identified as belonging to P.

If a remailer receives a request for indicator-key pairs naming another remailer - that is to say, for example, that if P receives the message:

$$\{\boxed{R,\ z,\ N_A}, \boxed{I_{Ak}, K_{Ak}, I_{Qj}, Q, \{E_Q\}_{K_{Qj}}}\}_{K_P}$$

it is being asked for indicator-key pairs which were generated by and are for use with the remailer R. If it has any spare it sends some of them, selecting them at random from its stock. It then removes them from its stock so that it does not use them itself.

If it does not have any labels for the remailer R it (if willing) starts to acquire some, so that if the enquirer asks again it will be able to answer in the positive. It does not reply until or unless it is asked again.

Without this mechanism there is a continuous chain of identification and linkage between the entity making a request for indicator-key pairs and them later returning with the same indicator-key pairs in a later request for remailing. With this mechanism a certain proportion of the labels seen by a remailer will not be being used for the benefit of the agent which originally requested them, so that the linkage between one set of labels used by an agent and another is diluted. This is especially true since a large proportion of the labels generated are used only for acquiring more labels.

The use of the labels' age to provide protection against the replay of old messages is another improvement over the public-key based protocol, which would need the introduction of globally synchronised clocks or perhaps some kind of additional challenge-response mechanism.

6.9. Anonymous certificates of location, and other more advanced protocols

The use of location information is not restricted to simple requests by enquirers for the location of a user, where the enquirer is happy to have the enquiry answered by a program running on the user's behalf and to trust the user's answer.

In some applications an enquirer may wish to be assured of a user's location using a mechanism which does not require them to trust the user not to lie. If instead they trust the transceiver infrastructure not to lie this can be achieved, and without revealing the user's location to anyone else, by the use of an 'anonymous certificate of location':

The enquirer U generates a nonce N_U and sends it with the rest of its request to the user's agent (at least encrypted so that it is only available to A, possibly even using the remailing infrastructure).

The user's agent A generates the following message and sends it to the transceiver infrastructure controller as a certificate request:

$$\{ \boxed{L, N_U, N_{Ai}, \{N_{Ai}\}_{K_{AB}}}, \boxed{I_{Ak}, K_{Ak}, I_{Qj}, Q, \{E_Q\}_{K_{Qj}}} \}_{K_S}$$

The controller transmits $\{N_{Ai}\}_{K_{AB}}$ at the location L. If it receives the reply N_{Ai} then it knows that the entity which issued the request for a certificate has an accomplice at L, presumably the agent's mobile device. It issues the certificate:

$$\{ L, N_U, T_S \}_{K_S^{-1}}$$

This is sent back to A via Q in the usual manner.

A forwards the certificate to U, who can see by the presence of the nonce N_U that someone who knew N_U had an accomplice near L recently. Since only U, A and S know N_U and we trust S anyway that means that A obtained the certificate, and has an accomplice, presumably A's mobile device, near L.

There are situations where the added delay imposed by the requirement for anonymity is not required or even desired. Suppose, for example, that the user wishes to use a computer system part of whose log on authentication requires their mobile device to be present while they log on. They present themselves to the terminal at L and identify themselves and type their conventional password. The computer system, acting as an enquirer U, sends a signed challenge directly to the user's agent A:

$$\{ E, L, N_U \}_{K_U^{-1}}$$

The user's agent forwards this to the transceiver controller as a request for a certificate, this time giving its plain address as a reply address:

$$\{ \boxed{L, N_U, N_{Ai}, \{N_{Ai}\}_{K_{AB}}}, \boxed{I_{Ak}, K_{Ak}, I_{Aj}, A, \{E_A\}_{K_{Aj}}} \}_{K_S}$$

The transceiver broadcasts $\{ N_{Ai} \}_{K_{AB}}$ at L, receives N_{Ai}, issues the certificate as above and sends it to A. A can forward the certificate directly to U and the user may log on.

There are many variations possible on similar and dissimilar themes.

If the situation requires anonymity as well as timeliness the solution above will not suffice. Any interaction between the user's agent and the mobile device is ruled out by the time taken to navigate the remailer network. This means that appropriate cryptographic material would have to be generated in advance and stored in the mobile device or at the location doing the access control.

The anonymous certificates described above are no good if the enquirer needs to show them to a third party, because the nonce N_U only has special meaning to the enquirer. This could be solved by having a trusted entity (the transceiver controller, perhaps, which must already be trusted to provide accurate location information) take on the role of enquirer and certify the result, or by having A act as enquirer and sign the result on its own recognisance.

In some circumstances an enquirer may wish to know who is at a particular location, rather than where a particular person is. This can be achieved by having the transceiver controller keep a record of address labels which it has recently seen broadcasted and acted on. When an enquirer wants to know who is at a location they ask the relevant transceiver controller to forward their enquiry (including the location and an address — anonymous or otherwise — for the enquirer) using each of its set of recent address labels. When the enquiry finally ends up at the users' agents it will not carry the binding nonce N_A since that was only in the portion of the data encrypted under K_{AB}; however, if the agent wishes it may reply giving its identity or just to register its presence.

The transceiver controller can estimate how many mobile devices there are at a particular address by measuring the mean density of transmissions, and publish this information. This will give the enquirer an estimate of how many people there are at the address of whom they would otherwise be unaware because they did not reply to the enquiry.

Additionally, agents could reply anonymously to acknowledge their presence without identifying themselves, and possibly even establish a dialogue with other agents at the same location without needing to know each other's identities. This can be useful to allow the enforcement of access control policies such as 'anyone who is in the same room as me can see my location tracking data in the computer as well as my physical presence' — though such a policy can be subverted if a rogue user leaves their mobile device behind in another user's office. In this latter case and with a security policy like the one described the mobile device would act as a kind of 'bug', allowing the owner of the device to find out information about the room without being there themselves!

6.10. Limitations and practicality

Problems will arise when the mobile device is out of touch for a long time and runs out of address labels. The usual tradeoff between security and functionality exists here: the mobile device can go off-air (or transmit much less frequently), until it is 'refilled' (perhaps by the user pressing a button to indicate that they're willing to have their location known), or it can reuse old labels in the hope that this will allow the agent to find it and send it some new ones.

There will be a certain vulnerability to traffic analysis. The key setup messages will provide cover traffic for the actual mobile device data, but they do tend to be rather large. In order to solve this problem it will be necessary to make explicit provision for cover traffic between remailers and to places where the mobile device might be.

The user's agent is in a good position to cause such cover traffic to be generated; it knows where the user often is, and can most easily make its cover traffic follow the patterns of real traffic closely. There would have to be a protocol mechanism for requesting cover traffic, so that the user's agent could cause a remailer or sensor controller to generate messages of a size and frequency similar to that of the location announcements from the mobile device.

Padding must be inserted in messages so that the exact length of a message going out of a remailer does not reveal which incoming message it is related to. Outgoing messages must be queued by remailers; each remailer must at regular intervals sort its outgoing message queue into lexical order and transmit all the messages; this avoids obvious correlation between the arrival time of message and the departure time of the same message on its next leg.

Long chains of remailers are not significantly more secure than shortish ones; the greatest vulnerability to traffic analysis and other effective ways of identifying the endpoints of the communication is at the start and end of the trail. Furthermore, if the level of traffic at any particular remailer becomes small it becomes useless, since the hiding achieved by its batching and sorting depends on other traffic to obscure the relationship between input and output messages.

Therefore, in a network with a large number of agents only some of them should act as (or be used as) remailers. Groups of users could agree to have one of their agents act as a remailer, and take turns, or could band together to run remailers themselves. In very large organisations the formal structures required to operate these remailers might best be handled by organised labour, to avoid the tendency to integrate them into the organisation's management and computing infrastructure, thereby reducing their trustworthiness.

In an organisation the size of ORL or the Cambridge Computer Laboratory, with a few hundred employees, certain staff might unofficially offer remailing services and be trusted and used by others; alternatively if the organisation has a strong group structure such as that in the Computer Laboratory each group (in this case a research group) could run its own remailer, and inter-group rivalry might reduce the risk of security breach by cooperation amongst remailer operators.

As will be shown in the next chapter, these protocols have proved difficult to subject to formal analysis.

Chapter 7. Formal analysis of protocols

In this chapter the need for a new formal protocol analysis tool for dealing with confidentiality is described. A tool is developed which can verify the preservation of secrets, and is applied to a number of example protocols. The application of this tool to the location-hiding protocol is discussed; there is still the lack of an ability to express confidentiality of relationship.

It would be good to be able to verify the correctness of the protocols proposed in chapter 6 using a formal logic.

Various approaches to formal verification have been proposed [Liebl93]. The most widely studied approach has been that of Burrows, Abadi and Needham (BAN) in [BAN89]. Unfortunately, a verification by BAN or one of its derivatives is not a sufficient condition for a protocol to fulfil the claims made for it in the logic.

This is due to the fact that BAN does not make any attempt to ensure the preservation of secrets, by modelling who ought to be allowed to have them; nor does it check that the messages sent by principals in the protocol are reasonable assertions for those principals to make given the rest of the protocol.

Other authors have favoured much more detailed analysis, for example in [Syverson90] of the semantics of messages, attaching fewer 'high-level' meanings to messages. These approaches tend to produce very long proofs, and sometimes have difficulty modelling the high-level meanings and interpretations involved.

Some authors (e.g. [Meadows94]) prefer to model the information and options available to attackers. These schemes suffer from the problem that it is often not clear that the model of the attacker is complete.

All is not lost; the BAN approach can be fixed. Its problem is that the protocol idealisation step can allow meanings to be attributed to messages which the sender of the message has no reason to believe, or messages which contain confidential information to be sent in the clear or encrypted using the wrong keys. These flaws allow the recipient of a message to come to believe a falsehood, and allow messages to be sent which leak secret information.

The possibility that secret information could be published was pointed out by Nessett in [Nessett90], where he presents a protocol with a deliberately introduced error which the BAN logic failed to spot. BAN et al note [BAN90] that this was because a key was sent in the clear when it should have been kept secret. They contend that since this action is not consistent with the assumption that the key is suitable for use it is not up to the logic to detect the mistake. However, though it is a simple example, this is one of the kinds of mistake that one might expect a formal logic to have exposed.

7.1. New logic

Below is presented a logic in which all messages sent must be shown to be publishable. This requires that if they are not encrypted they must be public knowledge. If they are encrypted the key used must be appropriate and messages implicitly signed by the use of symmetric or private keys must be shown to be believed by the sender.

The verification of conclusions and intermediate results is done separately for each principal. This ensures that the verification cannot unwittingly imply telepathic transfer between the principals, and allows each principal's trust and initial assumptions to be different.

The logic proposed here will not spot unimplementable protocols, and cannot as it stands handle non-monotonicity of knowledge.

It will also fail to expose contradictory assumptions, and doubtful practices such as session key chaining. However, some effort has been put towards the creation of a framework for justifying the assumptions made in use by those verifying protocols with the logic. This will hopefully make the use of false assumptions more difficult, particularly with respect to implementation details.

As we shall see, the logic proposed below is able to spot flaws in protocols which current logics do not; furthermore, its nature is that it will only verify protocols which are correct with respect to its assumptions, unlike other similar logics. However, due to an unusual feature of the protocols being proposed in the rest of this thesis it is unable to make a sensible verification of them.

It is hoped that its presentation here will nevertheless be useful in the construction of more conventional protocols, and in the development of more powerful protocol verification logics — particularly those which intend to verify only some correct protocols and no incorrect ones, rather than those which intend to verify all correct protocols at the cost of not spotting certain kinds of flaws in some incorrect protocols. It is also hoped that its presentation together with the reason for its failure in this instance will encourage better formal modelling of secrecy in general, and of secrecy of traffic flow and confidentiality of relationship in particular.

7.2. The logic's elements

The formal system that is the logic will now be described.

The formal portion of the logic consists of a syntax and simple type system for expressions, and rules for deriving expressions from previously derived or introduced ones. Expressions can composed from subexpressions using operators. There is a semi-formal linkage between the formal portion of the logic and the actual protocol which is to be analysed. Both the formal portion of the logic and this semi-formal linkage are described below.

When using the logic to analise a particular protocol the assumptions required of each principal by the protocol are first stated as expressions in the logic (as detailed below). Then it must be shown that, for each message α which the protocol requires a principal to send, the expression $\uparrow\alpha$ can be derived for that principal, from only the initial assumptions and the expressions in the logic corresponding to any messages which have been received by the relevant principal before that point in the protocol. It must also be shown that each belief which it is claimed is a result of the protocol has an expression in the protocol which can be derived from the initial assumptions and received messages.

7.2.1. Types of expressions

All expressions and subexpressions in the logic should be viewed informally as pieces of information. There are some particular types of information with additional characteristics; some operators apply to any kind of subexpression and others to only subexpressions with the correct type. General pieces of information are indicated with the Greek letters α, β, γ, δ, ε etc. A piece of information may be empty.

A subexpression may be a key of some kind. Keys are generally represented by the letter K, and are distinguished from each other using sub- and superscripts.

A subexpression may be a statement. Statements should be viewed informally as assertions. All axioms (assumptions about initial states, trust relationships, and messages received) and theorems proved as part of the analysis for a particular principal are statements. Statements are indicated by capital Greek letters whose form is distinct from Roman letters, for example Ω, Φ, Σ, Γ. When used merely as a piece of information a statement represents the information contained within its subexpressions. Kinds of statements not provided for in the logic may be introduced in assumptions, to model the conveyance of arbitrary information between principals – these statements should remain opaque to the logic except for the pattern-matching used for the $\rightarrow$ operator. Such statements are indicated by the use of English phrases in *cursive script* for the statement itself or as operators connecting other pieces of information.

A subexpression may be the name of a principal, or a set of zero or more principals. Principals are indicated by capital Roman letters (except where the specific letter is used for another purpose, such as K or T). Sets of principals are bold Roman letters such as $\mathbf{H}$. When used merely as a piece of information a set of principals or the name of a principal represents the information about the contents of the set or the identity of that principal.

A subexpression may be a timestamp. Timestamps are indicated by the letter T; if there are several they are distinguished using sub- and superscripts.

7.2.2. Meanings of symbols

The words in bold are to be interpreted as symbols whose use is governed by the rules of the logic, described below. A more compact representation which replaces the words with single-character symbols of various kinds is possible; however, the notation using words suggestive of the meaning of a symbol was chosen as it was felt to be easier to understand.

This is the Special Edition. It uses these symbols, because they are more compact and more pretty. Please consult the Standard Revised Edition for the version with words.

The types of the arguments to operators are indicated by the symbols used for the arguments.

7.2.2.1. Operators used to form statements

$\lozenge\ \alpha$	α is recent
$\#\ \alpha$	α is fresh and unpredictably generated
$\triangledown\ T$	T is a current timestamp
$\leftrightarrow K$	K is a 'good' symmetric key (i.e., works and is not weak or leaked)
$\rightleftharpoons K$	K is a 'good' public key (i.e., works and is not weak or leaked)
$\uparrow \alpha$	α may be said
$!\ \alpha$	A tautological statement; always true – this is used to carry information about inside statements without asserting anything
$\Omega \vdash \Phi$	If assumptions Ω are true then Φ is too
ʒ̸ α	α is not secret, and may be revealed to anyone
$S \rightarrow \Omega$	S is trustworthy with regard to statements matching the statement pattern Ω. Ω is a description of a set of statements, represented as a statement. Ω may contain some symbols not otherwise used in the analysis at hand, in which case a particular statement matches Ω if it has the right form and any subexpression(s) of the correct type(s) appear in place of the unbound symbol(s).
$A \propto \alpha$	A has knowledge of α
$\sim \alpha$	α has been seen as a message or part of one
$\approx \alpha$	α has been received in the clear
※ A	A exists at the moment, and has done something recently
$\alpha = \beta$	the pieces of information α and β are identical

7.2.2.2. Other operators and symbols

K^{-1}	The secret key corresponding to a public key K
ʒ α	The set of principals whose members are permitted to know the piece of information α and are trusted not to reveal it outside that set
H	The set of honest principals, who are trusted not to forge messages claiming to be from other people
I	Myself — the principal whose reasoning is being verified
Φ, Ψ	Composition — equivalent to $\Phi \wedge \Psi$ but with different precedence
$\wedge$ $\vee$ $\cup$ $\cap$ $\subset$ $\supset$ $\in$	Assorted logical and set operators. $\wedge$ $\vee$ take statements as operands and are statements; $\cup$ $\cap$ take sets as operands and are sets; $\subset$ $\supset$ take sets as operands and are statements; $\in$ takes a principal and a set of principals and is a statement.
$\{\alpha\}_K$	α encrypted using key K (a piece of information)
$A \models \Omega$	Used to indicate the sender of a message which has been given cryptographic integrity protection. This is a piece of information which appears inside encryption or signature in transmitted messages. Its appearance there serves to prevent reflection attacks, by identifying the sender explicitly.

Symmetric key encryption $\{\alpha\}_K$ with $\leftrightarrow K$ must provide both confidentiality and integrity. An ordinary block cipher used alone will not suffice — a MAC will also be required.

7.2.2.3. Operator precedence

Most tightly binding first:

( )
$\{\alpha\}_K$
\# ʒ $\leftrightarrow$ ※ ∇
$\cup$ $\cap$ $\propto$
! ʒ̸ $\sim$ $\approx$
$=$
$\models$ $\rightarrow$ $\uparrow$
$\wedge$
$\vee$
,
$\vdash$

7.2.3. Substantive rules

Rules for deriving expressions are presented in the form

$$\frac{\textit{condition}}{\textit{result}}$$

which states that *result* can be derived from *condition.* Rules may have several conditions separated by commas, meaning that the result can only be derived from all of the conditions together (all the conditions need to have been previously derived or introduced).

Things are recent if they are fresh or bear a current timestamp:

$$\frac{\#\,\alpha}{\Diamond\,\alpha} \qquad \frac{\triangledown\, T}{\Diamond\,(\triangledown\, T, \alpha\,)}$$

Content-free statements are true and believable:

$$\frac{}{!\,\alpha} \qquad \frac{}{A \rightarrow\ !\,\alpha}$$

Things sent in the clear aren't secret and are visible to us:

$$\frac{\approx \alpha}{\not{ʒ}\,\alpha} \qquad \frac{\approx \alpha}{\sim \alpha}$$

The distinction between $\approx$ and $\sim$ is that the former means we actually received the message in the clear and so it must be public, whereas the latter may have been buried inside some encrypted package.

Control over transmitted messages: $$\frac{\not{ʒ}\,\alpha}{\uparrow \alpha}$$

This is where the logic controls what may be sent. No message may be sent by a principal without it having been shown in the logic that it may, using this rule.

Use of symmetric key cryptography:

$$\frac{\Omega, \leftrightarrow K, ʒ\, K \subset ʒ\, \Omega, ʒ\, K \subset ʒ\, I}{\not{ʒ}\,\{\, I \models \Omega \}_K}$$

Control over the content of messages implicitly signed by the use of symmetric key cryptography is handled here, as well as the assurance of confidentiality. This is achieved by not making it possible to derive that an encrypted message is not secret and therefore publishable without using this rule. The requirements are that the statement to be asserted is believed to be true and that the key to be used is good and only available to a subset of the people to whom the information being sent should be made available.

Use of the secret key:

$$\frac{\Omega,\ \rightleftharpoons K}{\backepsilon\,\{\,I \models \Omega\}_{K^{-1}} = \backepsilon\,\Omega \cap \backepsilon\,I \cap \backepsilon\,K}$$

This rule merely assures truthfulness, since use of a secret half of an asymmetric keypair doesn't provide confidentiality. Therefore, if the key is good and the assertion is true, the signed version of the assertion may be shown to anyone who may see the assertion itself, the key and the name of the principal doing the signing.

Use of the public key:

$$\frac{\rightleftharpoons K, \backepsilon\,K^{-1} \subset \backepsilon\,\alpha,\ \not\backepsilon\,K}{\not\backepsilon\,\{\,\alpha\,\}_K}$$

Here we don't need to show that we believe the thing we are about to send, since the use of the public key doesn't imply signing. Instead, we simply ensure that the key is good and not secret and that the secret key is not available to anyone who shouldn't see the message. If this is the case then the encrypted message may be shown to anyone.

Message content (symmetric key):

$$\frac{\sim\{A \models (\,\Phi, \Omega, \Psi\,)\,\}_K\,,\ A \rightarrow \Omega\,,\ \leftrightarrow K\,,\ \backepsilon\,K \subset \mathbf{H} \cup \{IA\},\ \Diamond\,(\,\Phi, \Omega, \Psi\,) \vee\ \Diamond\,K}{\Omega,\ ※\,A\,,\ A \propto (\,\Omega, K\,)}$$

When we receive a message we must check that the alleged sender has authority to make the claim in question, that the key used is good and is only available to the real sender and to others who are trusted not to lie, and that the message is recent. When this has been confirmed we can believe the contents of the message, and also know that the sender exists and has access to the key and the message. (Note that Φ and/or Ψ may be empty.)

Message content (public key):

$$\frac{\sim \{ A \models (\Phi, \Omega, \Psi) \}_{K^{-1}},\ A \rightarrow \Omega,\ \rightleftharpoons K,\ \ni K^{-1} \subset \mathbf{H} \cup \{IA\},\ \Diamond (\Phi, \Omega, \Psi) \vee \Diamond K^{-1}}{\Omega,\ \divideontimes A,\ A \propto (\Omega, K^{-1})}$$

A similar situation applies to messages signed with the public key. (Again, Φ and/or Ψ may be empty.)

Decryption: $$\frac{\sim \{\alpha\}_K, \leftrightarrow K}{\sim \alpha} \qquad \frac{\sim \{\alpha\}_K, \rightleftharpoons K}{\sim \alpha}$$

We can see things we've decrypted. This allows an incoming public-key signed encrypted message to be decrypted and then processed further with the message content rule, for example.

Principals are allowed to see what they already possess: $$\frac{A \propto \Omega}{\ni \Omega \supset \{IA\}}$$

7.2.4. Rewriting rules

These rules allow statements in the logic to be rewritten. They specify what properties of composite messages and components of messages can be found from which properties of their components and containing messages (for example, if all of a compound message has been seen then each of its parts individually has also been seen):

Unary operators — replacing $\otimes$ with each of the operators $\#\ \Diamond\ \rightleftharpoons\ \leftrightarrow\ !\ \not\ni\ \sim\ \approx\ \divideontimes\ \ni$:

$$\frac{\sim \otimes \alpha}{\sim \alpha} \qquad \frac{\approx \otimes \alpha}{\approx \alpha} \qquad \frac{\not\ni \otimes \alpha}{\not\ni \alpha} \qquad \ni \otimes \alpha = \ni \alpha \qquad \frac{\# \alpha}{\# \otimes \alpha}$$

Binary operators — replacing $\otimes$ with each of the operators $\models\ \rightarrow\ ,\ \wedge\ \vee\ =\ \propto$:

$$\frac{\sim (\alpha \otimes \beta)}{\sim \alpha,\ \sim \beta} \qquad \frac{\approx (\alpha \otimes \beta)}{\approx \alpha,\ \approx \beta} \qquad \frac{\not\ni (\alpha \otimes \beta)}{\not\ni \alpha,\ \not\ni \beta} \qquad \ni (\alpha \otimes \beta) = \ni \alpha \cap \ni \beta$$

$$\frac{\# \alpha \vee \# \beta}{\# (\alpha \otimes \beta)}$$

$\not\ni \alpha$ is equivalent to $\ni \alpha = \{\text{all principals}\}$.

7.2.5. Assumption types

It is obviously important not to assume things that aren't reasonable considering the real situation. To assist in this all assumptions (axioms, in the logic) are categorised:

$\boxed{\mathrm{T}}$ Trust: list of people to believe. Usually statements of the forms $\ldots \subset \mathbf{H}, A \rightarrow \ldots$. These assumptions appear at the start of the analysis.

$\boxed{\mathrm{I}}$ Initial state: bootstrapping keys, their properties and those who are trusted with them. For example: $\leftrightarrow K$, $\backepsilon\ K = \ldots$, $\not\backepsilon\ \alpha$ (often required in order to be able to reveal the names of the principals to eavesdroppers). Again, these assumptions appear at the start of the analysis.

$\boxed{\mathrm{C}}$ Things created by the principal whose reasoning is being verified: session keys, nonces etc. and their properties. For example: $\leftrightarrow K$, $\backepsilon\,\alpha = \ldots$, $\#\,\alpha$, $\not\backepsilon\,\alpha$. These assumptions should appear in the analysis at the point at which the key or nonce is created.

$\boxed{\mathrm{R}}$ Message received. This gives the form and content of a message received. Principals and keys in the message must be checked against the same instances of the principals and keys existing in assumptions already made, and must be checked against instances of the same pieces of data elsewhere in the message. Timestamps ($\triangledown T$ in messages) are verified by introducing an assumption of the form $\boxed{\mathrm{R}}\ \triangledown\ T$ (implying of course a check by an implementation that T is a recent time); this allows the freshness of the message contents to be established and thus the meaning used.

The validity of $\boxed{\mathrm{T}}$ and $\boxed{\mathrm{I}}$ assumptions is the responsibility of the user of the protocol, including any external key exchange channel.

The validity of $\boxed{\mathrm{C}}$ and $\boxed{\mathrm{R}}$ assumptions is the responsibility of the protocol implementor, who must ensure that the keys, nonces and timestamps generated are good and check that messages received are really as expected.

7.3. Cautions regarding the use of the logic

The conclusions the logic reaches must be treated with some care, as they depend on the initial assumptions.

Trust and initial state assumptions ($\boxed{\mathrm{T}}$ and $\boxed{\mathrm{I}}$) should not contain trust information which is specific to the set of principals for which the protocol is being analysed (unless the protocol is only ever intended for use by that set). For example, when A is attempting to establish communication with B, A might have to use an assumption of the form $\{B\} \subset \mathbf{H}$. Assumptions of this type (those that are only valid in the particular context of the exchange in question) should be explicitly stated at all times, by placing them on the left hand side of the $\vdash$ symbol. This will ensure that they remain attached to any conclusions derived from them.

The contents of received messages are presented as assumptions in the logic, as they are obviously premises that the rest of the analysis will depend on. It is important that an implementation really does check that everything in received messages corresponds to that assumed, and doesn't continue the protocol run if it doesn't.

It is not expected that messages will be transmitted in a literal encoding of the logic. However, all the information that appears in a message in the logic must also appear in the actual transmitted message.

If a piece of information appears several times in the same part of a message (or is unambiguously implied by the key) used to encrypt or sign that part then it is acceptable to include only one copy of it (or none if it is implied by the key) in the actual representation of that part; this also relieves the receiver of the burden of checking that all the instances of the information are the same. Here a part of a message is a subexpression all of which is subjected to the same cryptographic processing with the same keys, according to the rules for the use of symmetric or private keys, or all to no cryptographic processing at all.

A message type code may be necessary to allow the receiver to distinguish between messages containing data of a similar form but with different meanings and so different idealisations in the protocol. Likewise an explicit indication of the sender of a message will be required if a symmetric or private key is shared between several parties and more than one of them ever uses it on similar outgoing messages.

The assumptions on the left hand side of the $\vdash$ operator can be quite lengthy, and may need to be abbreviated. Often they may be omitted completely in transmitted messages, if the messages in question contain the names of the principals and other information from which the assumptions can be divined, thus allowing the recipient to reconstruct the notional message.

7.4. Inability to verify all correct protocols, versus ability to 'verify' some incorrect ones.

The logic proposed here is sometimes unable to verify a protocol that is not broken. An example would be a modified Otway-Rees. The problem occurs when certain kinds of message chaining are used to avoid stating the names of principals and other such information explicitly in the protocol messages.

It may be possible to extend the logic to be able to verify these protocols, however the benefits are doubtful. Engineering robustness principles [AN96] suggest that explicit messages should be used instead of convoluted and hard-to-analyse chaining schemes.

7.5. Example — the Kerberos authentication protocol [kerberos87]

The Kerberos protocol is typically written as:

$$
\begin{aligned}
&A \textit{ sends to } S: \; A, B\\
&S \textit{ sends to } A: \; \{ T_S, L, K_{AB}, B, \{ T_S, L, K_{AB}, A \}_{K_{BS}} \}_{K_{AS}}\\
&A \textit{ sends to } B: \; \{ T_S, L, K_{AB}, A \}_{K_{BS}}, \{ A, T_A \}_{K_{AB}}\\
&B \textit{ sends to } A: \; \{ T_A + 1 \}_{K_{AB}}
\end{aligned}
$$

These messages are an encoding of the meanings:

A sends to *S*:

$$A, B$$

S sends to *A*:

$$\{ S \models (\triangledown T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, K_{AB} \textit{ has lifetime } \lambda, \; \# K_{AB}, \not\ni \omega) \}_{K_{BS}}$$

where ω is

$$\{ S \models (\triangledown T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, K_{AB} \textit{ has lifetime } \lambda, \; \# K_{AB}) \}_{K_{BS}}$$

A sends to *B*:

$$Q, \{ A \models \; ! T_A \}_{K_{AB}}, B$$

B sends to *A*:

$$\{ B \models \; ! T_A \}_{K_{AB}}$$

This will be explained more fully in the verification for each principal, below.

7.5.1. Verification for principal *A*

Initial trust assumptions:

We believe the server's and our own honesty in the use of keys, so that we do not doubt the origin of messages which claim to be from *B* and are signed by keys also held by us or by *S*:

$$\boxed{\mathrm{T}} \qquad \mathbf{H} \supset \{AS\}$$

We believe the server to be a competent authority for generating fresh session keys, determining who they are for and setting their expiry times:

$$\boxed{\mathrm{T}} \qquad S \rightarrow (\leftrightarrow K, \ni K = \{AXS\}, K \textit{ has lifetime } \mu, \; \# K)$$

We also believe the server if it tells us something is not secret. This is necessary because the protocol requires us to take something that the server sends to us encrypted under our session key, and publish it by sending it to B unencrypted.

$$\boxed{\mathrm{T}} \qquad S \rightarrow \not\ni \alpha$$

Initial state assumptions:

We initially share a good long-term symmetric key with the server:

$$\boxed{\mathrm{I}} \qquad \leftrightarrow K_{AS} \qquad \ni K_{AS} = \{AS\}$$

We must also be allowed to publish our own identity and that of the parties with whom we are communicating:

$$\boxed{\mathrm{I}} \qquad \not\ni A,\ \not\ni B,\ \not\ni S$$

First message:

Since A and B are public, the combination of the two must also be public (according to the binary rewriting rules, page 43, and the equivalence of $\not\ni \alpha$ with $\ni \alpha = \{$all principals $\}$). So,

$$\therefore \qquad \not\ni (A, B)$$
$$\therefore \qquad \uparrow (A, B)$$

... send this to S

Second message:

At this point S does some processing. We do not check S's decisionmaking here — that is analysed below. We simply expect to receive the right kind of response. In the actual implementation we wait for the message to arrive, and check that it corresponds to the message expected. Formally, the result of this check becomes a new axiom for the logic:

$$\boxed{\mathrm{R}} \qquad \approx \{S \models (\triangledown T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, K_{AB}\ \textit{has lifetime}\ \lambda,\ \#\ K_{AB,}\ \not\ni \omega\ )\}_{K_{AS}}$$
$$\boxed{\mathrm{R}} \qquad \triangledown T_S$$

Note that we must check the timestamp in the messsage — this takes the form of an $\boxed{\mathrm{R}}$ assumption in the logic stating that the timestamp we found in the message is current.

ω is an indigestible cryptographic lump. It corresponds to $\{S \models \Sigma\}_{K_{BS}}$ as sent by S (see the verification for S, page 51), but it is opaque to A.

We now wish to apply the symmetric key message content rule (page 42). This will require us to show that the encrypted portion is recent. We can do this with the timestamp rule (page 41):

$$\lozenge\ (\ \triangledown\ T_S, \leftrightarrow A_{AB}, \ni K_{AB} = \{ABS\}, K_{AB}\ \textit{has lifetime}\ \lambda,\ \#\ K_{AB},\ \not\ni\ \omega\ )$$

Now we can apply the message content rule, twice. First, consider:

Φ :	*empty*
Ω :	$\leftrightarrow K_{AB},\ \ni K_{AB} = \{ABS\}, K_{AB}\ \textit{has lifetime}\ \lambda,\ \#\ K_{AB}$
Ψ :	$\not\ni\ \omega$

Since (Φ, Ω, Ψ) is fresh, and $S \rightarrow \Omega$ (by our assumption about the server's competence to issue keys, with $K = K_{AB}$), and the key used K_{AS} is only known to members of **H**, we can believe Ω:

$$\therefore \qquad \leftrightarrow K_{AB}\ ,\ \ni K_{AB} = \{ABS\},\ K_{AB}\ \textit{has lifetime}\ \lambda,\ \ \#\ K_{AB}$$

With Ω equal to the Ψ above, Φ equal to the old Ω, and Ψ empty, we can derive:

$$\therefore \qquad \not\ni\ \omega$$

Now we know pretty much what we wanted — that we have a good fresh key with a known expiry time that no-one but A, B and S will see.

Third message:

However, we want to send B the session key, and something to check that B exists.

First we generate a timestamp, which (since we invent it) we can decide is not to be secret:

$$\begin{array}{ll} \boxed{\text{C}} & \not\ni\ T_A \\ \therefore & \ni T_A \supset \{S\} \end{array}$$

We wish to encrypt this for transmission to B, using K_{AB}. We must use the rule for the use of the symmetric key (page 41). Of the conditions that must be satisfied, we already believe the statement we are about to send, and that the key is good. Furthermore, since both the message and our identity are public, the condition that the key be at least as secret as the message is also satisfied:

$$\begin{array}{ll} \therefore & \not\ni\ (\ \{\ A \models\ \ni T_A \supset \{\ S\}\}_{K_{AB}}\ ) \\ \therefore & \not\ni\ (\ \omega\ ,\ \{\ A \models\ \ni T_A \supset \{\ S\}\}_{K_{AB}}\ ) \\ \therefore & \uparrow (\ \omega\ ,\ \{\ A \models\ \ni T_A \supset \{\ S\}\}_{K_{AB}}\ ) \end{array}$$

... send this to B

Fourth message:

We expect B to reply as follows:

$$\boxed{\text{R}} \qquad \approx \{\ B \models\ \ !\ T_A\ \}_{K_{AB}}$$

Once again, we must use the symmetric-key decryption rule. We've seen an encrypted message, allegedly from someone who does have authority over the statement in question — according to the rules about content-free statements every principal is authoritative for them (page 41). The key is good. The people who possess the key are $\{ABS\}$; A and S are trusted not to abuse it (are in **H**), and B is the claimed sender, so the key is only available to the right principals. Finally, the key is recent, which satisfies the recency criterion:

$$\therefore \qquad !\,T_A\,,\ \text{※}\,B,\ B \propto (\,!\,T_S, K_{AB}\,)$$

We are not interested in B's possession of the timestamp, or in the tautology about it. However, we've now shown that:

$$\begin{array}{l} \leftrightarrow K_{AB} \\ \ni K_{AB} = \{ABS\} \\ K_{AB}\ \textit{has lifetime}\ \lambda \\ \#\,K_{AB} \\ \text{※}\,B \\ B \propto (\ldots, K_{AB}) \end{array}$$

I.e., we know that we share a good, fresh symmetric key K_{AB} with lifetime λ with A, B and S (whom we trust not to abuse it), and that B exists and is in possession of this key.

7.5.2. Verification for principal *B*

Unsurprisingly, the verification for B is fairly similar to that for A in many respects.

Initial trust assumptions:

As with A, we believe the server's and our own honesty in the use of keys:

$$\boxed{\text{T}} \qquad \mathbf{H} \supset \{BS\}$$

As with A, we believe the server to be a competent authority for generating fresh session keys and telling us their properties:

$$\boxed{\text{T}} \qquad S \rightarrow (\leftrightarrow K, \ni K = \{BSX\}, K\ \textit{has lifetime}\ \mu,\ \#\,K)$$

B does not need to believe the server's claim that a message is not secret, because it does not decrypt something supplied by the server and forward it elsewhere.

However, it does need to know who can see the timestamp transmitted in the third message. This is dealt with by assuming that any counterparty can tell us that something can be shared with the server:

$$\boxed{\text{T}} \qquad D \rightarrow \ni \delta \supset \{S\}$$

Initial state assumptions:

We initially share a good long-term symmetric key with the server:

$$\boxed{\mathrm{I}} \qquad \leftrightarrow K_{BS} \qquad \ni K_{BS} = \{BS\}$$

B does not need to assume that the identities of the parties can be made public, because it receives these identities during the protocol and can derive the necessary information.

Third message (first message seen by B):

The first thing B knows is that it receives:

$$\begin{array}{ll} \boxed{\mathrm{R}} & \{ S \models (\triangledown T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, \\ & \quad K_{AB} \textit{ has lifetime } \lambda,\ \# K_{AB}) \}_{K_{BS}} \\ \boxed{\mathrm{R}} & \{ A \models \ni T_A \supset \{S\} \}_{K_{AB}} \end{array}$$

This is two messages combined, and we deal with them separately. First, consider the message encrypted under K_{BS}. If we check the timestamp, this satisfies the conditions in the symmetric key content rule (page 42):

$$\begin{array}{ll} \boxed{\mathrm{R}} & \triangledown T_S \\ \therefore & \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, K_{AB} \textit{ has lifetime } \lambda,\ \# K_{AB},\ \divideontimes A,\ A \propto (\ldots, K_{AB}) \end{array}$$

So, by unpacking $\{\ldots\}_{K_{BS}}$ we have found out all the information we need about the key: it is good, fresh, is a secret shared amongst $\{ABS\}$, and has a lifetime λ.

We can use it to decrypt the second half of the received message, once again using the symmetric key message content rule. We rely on our assumption that anyone can tell us we can share a secret with the server:

$$\begin{array}{ll} \therefore & \ni T_A \supset \{S\},\ \divideontimes A,\ A \propto (\ldots, K_{AB}), \ni T_A \supset \{AB\} \\ \therefore & \ni T_A \supset \{ABS\} \\ \therefore & \ni (\ !\ T_A\ ) \supset \{ABS\} \end{array}$$

Fourth message:

We want to send the final message to A, to confirm that we have the key. Since the set of people amongst whom T_A is a secret definitely includes A, S and ourselves, we can safely send it encrypted under a key known only to those people:

$$\begin{array}{ll} \therefore & \not\ni \{ B \models\ !\ T_A \}_{K_{AB}} \\ \therefore & \uparrow \{ B \models\ !\ T_A \}_{K_{AB}} \\ & \textit{... send this to A} \end{array}$$

We therefore send A the reply which is expected.

7.5.3. Verification for principal S

Initial assumptions:

The server doesn't need to start out believing anything except that it shares good keys with A and B, and it does not need to trust either of them:

$$\boxed{\mathrm{I}} \qquad \leftrightarrow K_{AS}\,, \qquad \ni K_{AS} = \{AS\}$$
$$\boxed{\mathrm{I}} \qquad \leftrightarrow K_{BS}\,, \qquad \ni K_{BS} = \{BS\}$$

It does need to be willing to tell people its own identity:

$$\boxed{\mathrm{I}} \qquad \not\ni S$$

First message and its reply:

The server receives from A:

$$\boxed{\mathrm{R}} \qquad \approx (A, B)$$

Since it received this in the clear, it knows that it isn't secret:

$$\therefore \qquad \not\ni A\,, \qquad \not\ni B$$

It creates the key K_{AB}, with expiry time L, and the timestamp T_S:

$$\boxed{\mathrm{C}} \qquad \triangledown\, T_S\,,\ \not\ni T_S\,,\ \not\ni \lambda$$
$$\boxed{\mathrm{C}} \qquad \leftrightarrow K_{AB}\,, \ni K_{AB} = \{ABS\}$$
$$\boxed{\mathrm{C}} \qquad K_{AB}\ \textit{has lifetime}\ \lambda\,,\ \#\, K_{AB}$$

Compose most of these statements and we get:

$$\therefore \qquad \triangledown\, T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}, K_{AB}\ \textit{has lifetime}\ \lambda,\ \#\, K_{AB}$$

Call that statement Σ. From the rules about composition we have:

$$\begin{array}{ll} & \ni \Sigma = \ni\ \triangledown\, T_S \cap \ni K_{AB} \cap \ni \lambda \cap \ni ABS \\ \therefore & \ni \Sigma = \ni K_{AB} \\ \therefore & \ni \Sigma = \{ABS\} \end{array}$$

We wish to tell this information to both A and B. We use the rule about the use of the symmetric key (page 41) twice, once for each client. First, we use it for B:

$$\therefore \qquad \not\ni \{\, S \models \Sigma \,\}_{K_{BS}}$$

We now have the cryptographic package that will eventually get to B.

We compose another large message, with practically the same content — the difference is that A needs to know that it can send the package above on to B. So, we construct the new statement:

$$\therefore \qquad \Sigma\,,\ \not{\text{ɜ}}\,\{\, S \models \Sigma \,\}_{K_{BS}}$$

Who can see this ? Just the same people as can see Σ, because the rest is public:

$$\text{ɜ}\,(\,\Sigma\,,\ \not{\text{ɜ}}\,\{\, S \models \Sigma \,\}_{K_{BS}}\,) = \text{ɜ}\,\Sigma = \{ABS\}$$

So, we can encrypt it for *A*, and send it to them:

$$\begin{array}{ll} \therefore & \not{\text{ɜ}}\,\{\, S \models (\,\Sigma\,,\ \not{\text{ɜ}}\,\{\, S \models \Sigma \,\}_{K_{BS}})\,\}_{K_{AS}} \\ \therefore & \uparrow\{\, S \models (\,\Sigma\,,\ \not{\text{ɜ}}\,\{\, S \models \Sigma \,\}_{K_{BS}})\,\}_{K_{AS}} \\ & \qquad\qquad \textit{... send this to A} \end{array}$$

That is the end of *S*'s involvement in the protocol; it doesn't end up believing anything very useful — its purpose is to create a key for *A* and *B* and to arrange that they can have reason to believe in its usefulness.

Note that we are using Σ to simplify the analysis only. In the actual protocol all the relevant components are transmitted in full, though the encoding is up to the implementor.

7.6. Example — public-key variant of Needham-Schroeder

In [NS78] Roger Needham and Michael Schroeder present (amongst other protocols) a public key authentication protocol, which can be represented as:

$$\begin{array}{l} \textit{A sends to S}:\ A, B \\ \textit{S sends to A}:\ \{\, K_B\,, B\,\}_{K_S^{-1}} \\ \textit{A sends to B}:\ \{\, N_A\,, A\,\}_{K_B} \\ \textit{B sends to S}:\ B, A \\ \textit{S sends to B}:\ \{\, K_A\,, A\,\}_{K_S^{-1}} \\ \textit{B sends to A}:\ \{\, N_A\,, N_B\,\}_{K_A} \\ \textit{A sends to B}:\ \{\, N_B\,\}_{K_B} \end{array}$$

This representation is from [BAN89].

In order to attempt verification of this protocol we shall consider the following putative idealised meanings for the messages:

$$\begin{array}{l} \textit{A sends to S}:\ A, B \\ \textit{S sends to A}:\ \{\, S \models\ (\rightleftharpoons K_A\,,\ \text{ɜ}\,K^{-1}{}_A = \{A\}\,)\,\}_{K_S^{-1}} \\ \textit{A sends to B}:\ \{\, N_A\,\}_{K_B} \\ \textit{B sends to S}:\ B, A \\ \textit{S sends to B}:\ \{\, S \models\ (\rightleftharpoons K_B\,,\ \text{ɜ}\,K^{-1}{}_B = \{B\}\,)\,\}_{K_S^{-1}} \\ \textit{B sends to A}:\ \{\, !\,N_A,\ !\,N_B\,\}_{K_B} \end{array}$$

$$A \textit{ sends to } B: \{ N_B \}_{K_B}$$

However, A and B will not be able to apply the public key message content rule without showing that either the signing key K_S^{-1} or the message (which contains K_A and A's name) is fresh. This corresponds to the weakness discussed in [BAN89, section 6]— the principals do not have any reason to believe that these messages are fresh.

If we ignore this problem (for example, by introducing the freshness of K_S^{-1} as an assumption) we are still left with a number of problems, primarily due to the use of decryption by a principal as a form of verification of identity. Rather than having the principals A and B sign each others' nonces, the protocol has them encrypt them.

In order to verify the protocol signatures would have to be used instead of encryption, with the nonces being public. Furthermore, the principals would have to identify themselves as they exchange nonces, thus preventing an attacks which exchange the messages $\{ N_A \}_{K_B}$ and $\{ N_B \}_{K_B}$.

Such changes to the protocol, made to make it verifiable with the logic, would also prevent attacks such as that by Lowe [Lowe95] in which an attacker uses a legitimate principal as an oracle for decryption.

7.7. Example — wide-mouthed frog

In [AN96, example 11.2] Martin Abadi and Roger Needham consider as an example the wide-mouthed frog protocol:

$$A \textit{ sends to } S: A, \{ T_A, B, K_{AB} \}_{K_{AS}}$$
$$S \textit{ sends to } B: \{ T_S, A, K_{AB} \}_{K_{BS}}$$

Let us suppose that the intended meanings of these messages were as follows:

A sends to S:

$$A, \{ A \models (\triangledown T_A, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}) \}_{K_{AS}}$$

S sends to B:

$$\{ S \models (\triangledown T_S, \leftrightarrow K_{AB}, \ni K_{AB} = \{ABS\}) \}_{K_{BS}}$$

This protocol clearly does not convince A of B's presence. Despite this with appropriate assumptions (for example, B must assume that A is trusted to generate keys, and S must believe that anyone can do so), the verification can be achieved such that B is convinced that the key has the appropriate properties.

However, the idealised messages do not match the unidealised ones. In section 7.3 we say that everything that is in the idealised messages must also be in the actual transmitted ones. In this protocol the names of A and S appear in the first idealised messages, but not in the unidealised ones; likewise B and S in the second. The fact that one of A or S (respectively, B or S) is meant can be inferred from the use of K_{AS} or K_{BS} , but this does not say which. Therefore an additional indication would have to be added to distinguish the two messages.

This additional indication does indeed prevent an attack: an attacker might otherwise forward S's own message to B back to S, with B's name on the outside of the encryption. S would think that the message was from B, and send it's 'response' to A. The attacker can then do the same with that response. By a series of such manipulations an attacker could keep a session key K_{AS} valid well past its original lifetime, each bounce extending the timeout.

7.8. Analysis of remailer protocols

The remailing protocol considered is the public-key variant above (exactly which version we consider is not particularly important here).

We shall work with the un-idealised protocol (i.e., use notation that doesn't specify the meanings of the messages), because it is precisely the determination of these meanings that is problematic.

7.8.1. Identification of principals in messages

Each entity would have to include its identity in its outgoing messages (the rules for the use of private and symmetric keys require this). This appears problematic because the whole point of the protocol is to hide the indentities of the main participants (A and B) from the other principals (S and the remailers P, Q, etc.).

This could be solved in one of two ways, both of which would require a modification to the logic to weaken the requirement for principals to identify themselves. This requirement has two main purposes: firstly, it serves to prevent a principal from acting on its own messages, should an attacker play them back to it; secondly, it determines, during the verification, which $\rightarrow$ assumptions are relevant to whether to believe the contents of a particular message.

If we remove the requirement for identification of principals completely then we have to introduce two additional constraints. Firstly, the implementation would have to be able to recognise replayed copies of its own messages in all cases, and discard them; this achieves the first of the two aims of the identification. Secondly, the logic's rules would have to be modified so that the belief of a statement in a message is done by directly comparing the set of principals who are authoritative for a statement with the set of principals who have access to the key, rather than as is currently the case by determining the putative sender using the set of principals with access to the key and the set of honest principals $\mathbf{H}$ and then checking for a $\rightarrow$ assumption for the putative sender; the analysis would then require assumptions about the authoritativeness of principals who were previously explicitly trusted merely not to impersonate the authoritative ones.

Alternatively, the rules could be relaxed so that I, the identity of the sender, could be present in the message as a psuedonym rather than as an actual identity. If the psuedonym is required to be unique (either by virtue of being fairly long and randomly generated, or by virtue of having been centrally assigned) this would serve the first purpose of identification, since a principal's own messages would be recognisable as such. The second purpose would also be served: there would be no way for a recipient of a message to know which set of $\rightarrow$ assumptions to use unless it the psuedonym was meaningful to it; otherwise only $\rightarrow$ assumptions which do not specify a particular principal as the authority would be useable.

Either of these solutions would satisfy the requirement for anonymity on the part of A and B in the instant protocols. However, as we shall see in the next section, there is another problem which is more difficult to solve.

7.8.2. Public nature of endpoints' identities; confidentiality of relationship

In order for the remailer P to be able to send its outgoing message:

$$\boxed{\{E_Q\}_{K_Q}}, \boxed{\{N_{B,P}\}_{K_{AB}}, \{D\}_{\{f(N_{B,P})\}_{K_{AB}}}}$$

it needs to know that $\{N_{B,P}\}_{K_{AB}}$ and $\{E_Q\}_{K_Q}$ are not secret (as per the rule 'control over transmitted messages').

It might be possible to have the mobile device, which created $\{...\}_{K_P}$, tell the remailer P that the data isn't secret, and even to frame a sensible set of assumptions that will allow the remailer P to believe this despite the fact that it doesn't know who the sender is. This might entail an additional rule, for instance one stating that there is no harm in believing an assertion that a piece of information is not secret when made by someone who already possesses the information.

However, in order for B to be able to assert this in its message to P, it needs to know itself that it is true ('use of symmetric key cryptography'). There isn't anywhere this idea could come from, so we would end up introducing $\not\ni \{N_{B,P}\}_{K_{AB}}$ and so forth as assumptions.

However, this is undesirable — this data really is secret, in the sense that we do mind when and where it gets published. If it weren't secret we could do away with the cryptography altogether.

A similar problem occurs with D, which B must believe to be public knowledge when it sends the data, and which (because it has been seen in a public message) P can also assume to be public data ('things sent in the clear aren't secret'). So we're left with trying to figure out a way to express the fact that P needs to encrypt the data D *when it sends it with the* *addressing info* to Q. As the designers of the protocol, we can see that the cryptography is there to prevent an attacker from observing the connection with the data D, but there is no way to express this in the logic as it stands.

The importance of the connection between pieces of information, rather than of the pieces in isolation, is a common feature of identity-hiding protocols rather as opposed to traditional key exchange and/or authentication protocols.

So, though the logic proposed above may be useful for the verification of those traditional protocols it is not suitable for the problem at hand. The author has not encountered any other logic which has a good formalism to express this ‘relationship-hiding’. Without such a formalism it will be difficult to carry out a formal analysis of the correctness of the protocols proposed here.

More work on the formal expression of relationship hiding is required; in the meantime the logic presented above is at least able to deal with confidentiality, and may provide a framework for future enhancements dealing with confidentiality of relationships.

Chapter 8. Design of sample implementation

The core of the system which has been presented — the mobile entity, the remailer and the core functions of the user's agent — has been implemented. This chapter will describe details of the implemtation; the results of tests of it will be presented in the next chapter.

The programs, written for UNIX, use standard cryptographic primitives and communicate with each other with TCP/IP for simplicity. The remailer and user's agent have been integrated into a single program, and the agent portion delegates access control decisions to an external program.

8.1. Badge hardware

The existing Active Badge technology developed at ORL uses only very simple processors with a few tens bytes of RAM. At the time of its development (around 1992) this was as much as could be provided and powered in an object small and light enough to be worn on clothing (more than a few tens of grams in a package a few inches square is considered unacceptable).

Staff at ORL have indicated to the author that were the badge system to have been designed today it would probably have included a reasonably powerful but low power modern microprocessor such as an ARM and at least several hundreds of kilobytes of RAM. Therefore, to avoid restrictions from the older badge technology, a laptop computer connected to a modified badge wall sensor was used to simulate the badge.

The author was informed that the costs in terms of power consumption of infra-red communications have not changed significantly, and that this was now the chief problem with extending badge functionality within the weight and size restrictions.

The existing infra-red badge communications infrastructure was adequate; it supports messages of up to 255 bytes at a data link rate of 9600 bits per second.

8.2. Cryptography

The cryptographic algorithms used are the triple-DES block cipher and the RSA asymmetric cryptosystem [RSA78]. RSA is used to encrypt a randomly generated per-message symmetric key, and the actual message encrypted using that session key. This avoids the pitfalls with the direct use of RSA [PP89].

Where the formalised protocol specifies symmetric key encryption (and where symmetric key encryption is required for the data of public-key encrypted message, so that the public key is used to send a per-message symmetric key) it requires both confidentiality and integrity. Therefore, when symmetric key encryption is to be performed two keys derived from the specified key are used to generate one for confidentiality and one for MAC.

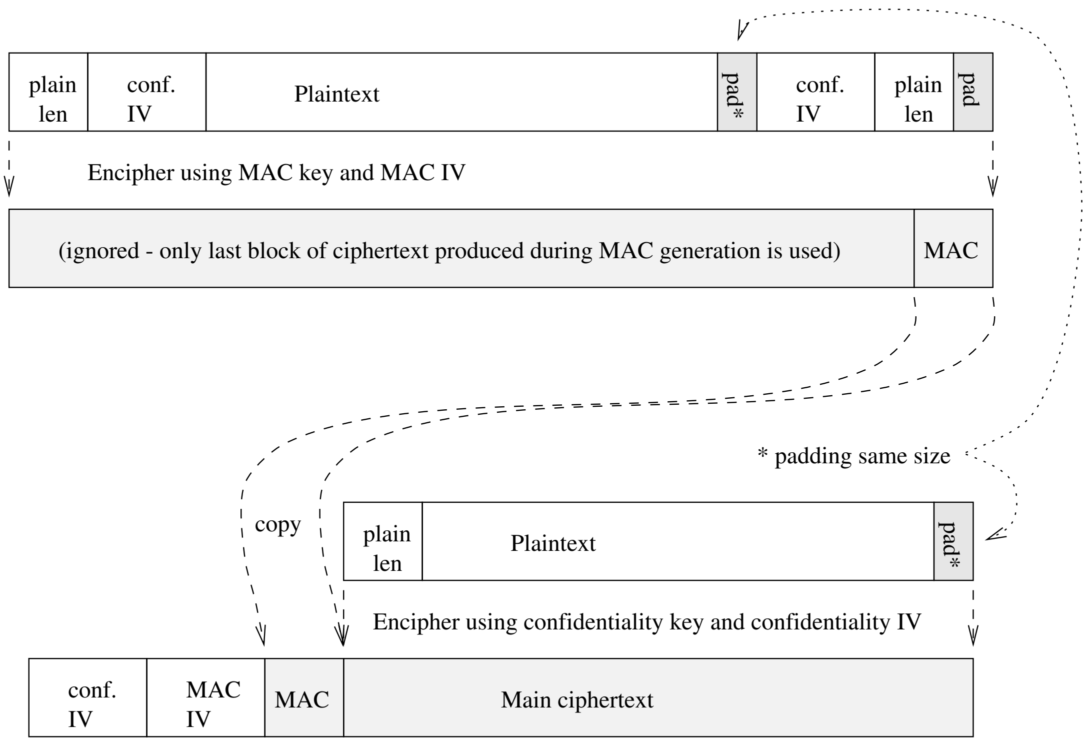
The length of the plaintext, as a 16-bit integer, is prepended to the plaintext, and the result is padded to a block boundary with zeroes. This text is enciphered using the confidentiality DES key and a random IV. This is the bulk of the ciphertext.

The IV which was used is prepended to the plaintext, and the plaintext length is prepended to that. The same amount of padding as was used earlier is appended, followed by another copy of the confidentiality IV and the plaintext length and more zero padding to bring the whole to a block boundary. This is enciphered using the MAC DES key and another random IV. All but the last output block is discarded; that last block is the MAC.

The whole ciphertext message consists of the confidentiality IV, the MAC IV, the MAC itself, and the encrypted ciphertext mentioned earlier. Figure 8.1 contains a diagram of the construction of symmetric key encrypted data.

When it is necessary to derive one or more keys from another key the message described below is enciphered using the master key and the final portion as long as a symmetric key is taken to be the derived key. The message to be enciphered is the concatenation of a piece of zero-terminated ASCII text which differs for each key to be derived, and enough padding that the result is a whole number of blocks and just long enough so that the first output block from which any of the resulting key material is taken corresponds to the last non-padding input block. The IV used is all-zeroes.

Figure 8.1. Use of symmetric key encryption for confidentiality and integrity



When a one-off symmetric key is required a key is derived from a one-off key generation key, using the procedure above, except that a 32-bit counter is included after the zero byte which terminates the ASCII text message.

8.3. Formats of transmitted messages

The messages between entities are packed binary data. Integers are presented in network order (big-endian). Field and message lengths are represented as 16-bit quantities.

There is a type byte at the start of each message, and inside the payload of an encrypted message, which indicates its type and format. However, to keep the message size down there is no indication of which cryptographic algorithms and message format versions are in use, so that a version using different cryptographic algorithms would have to use different link- and network-level protocol identifiers or port numbers to distinguish it from the current system.

Messages do not contain any padding to even locations; where required by cryptographic algorithms they are padded to block boundaries (see the description above for details of the way the cryptographic algorithms are used and how the padding is done).

There is provision for padding messages to hide their exact length; this is done separately, at a higher level than the cryptographic primitives.

8.4. Communication between entities

To simplify the implementation messages are sent from one entity to another via TCP/IP connections. This transport service provides reliable message delivery except in the case of deliberate attack or network failure.

An attacker can with a small amount of effort sabotage a TCP stream by causing it to appear to either end to have been closed by the other, and/or inject false data. However, since cryptographic mechanisms are used to ensure the confidentiality and integrity of data this is only a denial-of-service attack, to which the whole design is not resistant in any case.

Agents' network addresses are tuples consisting of an IP address and a TCP port; identification of agents to ensure that the right agent is being used is accomplished using the cryptographic protocols. Each TCP/IP connection is used only unidirectionally; the entity which did an active open (the 'client') uses it to transmit messages to the entity which did the passive open (the 'server'), and many messages may be sent down a single connection, which may be closed or left open when it is idle. Framing is accomplished by preceding each message with a 4-byte length field.

8.5. Operating system and programming language

The implemented system is targeted at UNIX and clones, since this provides a rich and reasonably stable development environment and provides straightforward access to multiprocessing and networking.

The system is written in C since that is the ‘native language’ of UNIX. It is particularly suited to the bit- and byte-wise manipulations required for the construction of messages and for cryptographic operations. It also allows the easy integration and reuse of existing library code, for example for cryptography and run-time programmability.

8.6. Overview of agent and remailer design

The user’s agent, remailers and transceiver controllers carry out very many similar functions, and so will share a lot of implementation. Additionally, it is advantageous to hide whether a particular message addressed to an entity will be processed by a remailer or by an agent. For these reasons the agent, remailer and transceiver controller are implemented as the same program.

The remailer/agent consists of two parts. The main portion of the remailer is the core of the system. It handles the cryptography; keeps all the keys, interprets requests for information from other entities, keeps track of the badge, stores information about ongoing queries, and performs the relaying and queueing required by its role as a remailer.

In order to decide whether or not to answer a particular query addressed to it as an agent, the core program asks the other part of the remailer/agent, a decision program provided by the user. This decision program can itself ask questions via the core program if it wants to, or do anything else the user feels is appropriate before answering. All the user’s policy is implemented by the decision program.

8.7. Internal design of core program

The main body of the remailer core repeatedly reads messages from the network and from its agent decision program, and deals with each message in turn. Outgoing messages are queued and sent in batches, sorted lexically by visible recipient and content to hide arrival and processing order.

Since it serves as the core of the agent too it communicates with other remailers to set up anonymous paths to itself, creating and maintaining a store of address labels for use by the mobile device and transmitting them to it periodically.

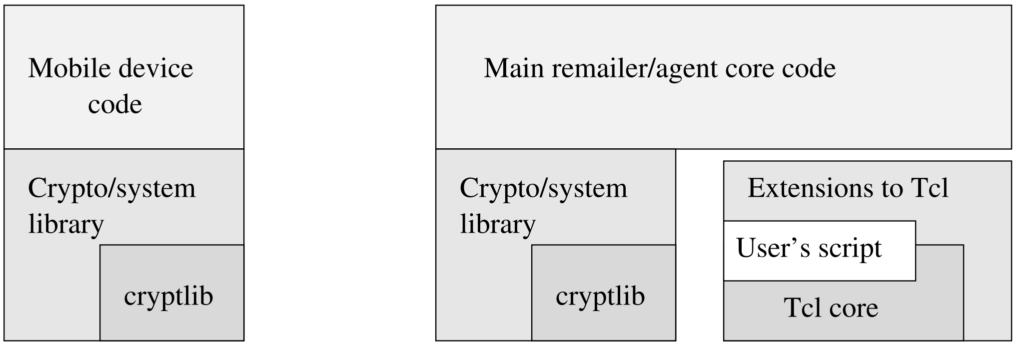
It keeps a record of each incoming query, so that it can reply appropriately (with the correct sequence numbers and addresses) when the decision program gives its response to the query.

The core program makes use of an I/O and scheduling library. This is used for communication with other entities, presenting a mostly-reliable unsecured datagram service to the body of the remailer, and a simple timeout despatch mechanism which invokes a callback when a certain time has elapsed.

It also makes use of a cryptographic library which provides asymmetric and symmetric key encryption, decryption, signing and verification. This library generates derived keys for use in MAC and confidentiality algorithms, and generates one-time symmetric keys for use with public key cryptography; it provides high-level cryptographic services. The interface to this library permits the algorithms to be replaced without affecting the operation of the program as a whole.

The triple-DES implementation used by the high-level cryptography library is that in Peter Gutmann's cryptlib [Gutmann97], which was written by Eric Young.

Figure 8.2. Program structures



8.8. Interface between core and decision program

The decision program is invoked as a subprocess by the agent; it communicates with the agent through a pair of pipes set up by the agent to be on its standard input and standard output.

The decision program receives questions (queries that the agent is considering answering) on standard input, and sends responses (decisions about which information to give out) to its standard output.

It should keep reading queries and answering them; it must be capable of handling many queries in each invocation. The query and response formats contain sequence numbers, so that the decision program may answer them out of order. This is useful (for example) if it wishes to make a (possibly time-consuming) enquiry itself before answering a particular query.

The agent will send both a request and a ‘suggested response’ containing the information requested by the enquirer, as the agent believes it to be; the decision program is expected to return the response it feels appropriate (which might be the ‘suggested’ response or something else).

Chapter 9. Performance tests of implemented system

The core of the system described in the previous chapter was implemented and tested under various conditions and with a view to determining its behaviour on startup, when the user's location changes, and when the packet loss levels are high or very asymmetric.

This chapter will present the results of those tests.

9.1. Testing environment

To facilitate testing the broadcast link between the mobile device and the various transceivers was represented by a simulation which allowed the probabilities of messages being received (i.e., the packet loss rate) to be adjusted independently in each direction and allowed the simulated location to be changed at will.

To simplify the data collection and reduction the testing scheme contained only one remailer/agent and one mobile device. The remailer was programmed to think of itself as another remailer that it would like to use, so that messages would pass through this remailer several times, and this remailer/agent was required to ask itself for indicator-key pairs just as if it were talking to another entity.

9.2. Parameters used during testing

The system as a whole has a number of tuneable parameters which will significantly affect its quantitative behaviour. During initial testing reasonable values for these parameters were established, and they were then kept constant for the remainder of the tests.

All the timing parameters described here were halved (and frequency or time density parameters doubled) for the actual tests to reduce the amount of real elapsed time taken by the testing. All the times discussed in the rest of this chapter are in thus reported in a 'virtual time' which elapses at twice the rate of real time.

The remailer/agent would queue outgoing messages and empty its queue in lexical order every 30 seconds.

The remailer would attempt to maintain a stock of at least 2000 indicator-key pairs for each of the other remailers it might wish to use (in the tests described below this means only itself). If its stock was below this and it was about to despatch queued outgoing messages (i.e., every 30 seconds) it would add to its queue 10 distinct requests for indicator-key pairs, each requesting 7 pairs. These requests would each be sent separately using the 'reverse' public key remailing scheme, and the return labels in the requests would cause the replies to come through several remailers on the way back. If the remailer had less than 750 indicator-key pairs for any other remailer it would instead send 15 requests for 10 keys.

The remailer was configured never to send more than 30 indicator-key pairs regardless of how many were requested, but this limit was not reached since it would never ask for more.

The remailer/agent would construct anonymous address labels so as to go through three remailers before arriving back at itself (the ultimate destination). Thus every label used up three indicator-key pairs. The agent would choose the three remailers randomly from its approved list, but in the actual testing scenario this list contained only one entry, itself.

Whenever the agent received a claimed location announcement from the mobile device it would send four address labels to the location in the claim, provided that the mobile device said it was more than two labels short of full. The agent would send two further batches of four labels, at intervals of 60 seconds.

The mobile device would transmit its location with a mean density of one transmission every 30 seconds. It would attempt to keep a stock of address labels sufficient for four hours' running without needing to receive any further supplies or to reuse old labels - thus, it would quote the number of labels it would like, in its messages to the agent, as 480 minus the number of labels it possessed.

If it ever found itself wanting to reuse a label which had been used more than 20 times previously it would discard it and hope it had a better one, unless that was its only remaining label. This is a serious error condition of which the user should be made aware (though it also occurs during bootstrapping).

When constructing a 'reverse' public-key-encrypted path the agent would start with a single hop and repeatedly add another step with probability $\frac{2}{3}$.

The measurements below do not include the effects of generating or receiving cover traffic.

9.3. Behaviour on startup

When started up, the mobile device has only one address label, which it has created itself. This label is not at all anonymous, and is used merely for bootstrapping. It is discarded as soon as some real labels are available.

Likewise, the remailer/agent starts off without any indicator-key pairs. It must request some from the other remailers it is to use, and even then the full confidentiality of location is not available until its association with the indicator-key pairs it is using is sufficiently diluted by exchange with other remailers.

Figure A.1 (page 80) shows the number of fresh address labels held by the mobile device and the number of indicator-key pairs held by the agent as functions of time after startup during a typical run where the user's location was kept constant and the probability of message loss was zero. It covers the first 6000 seconds (100 minutes) after startup.

Figure A.2 (page 81) shows a detail of the total number of address labels and the number of fresh labels held by the mobile device in the first ten minutes after startup.

As can be seen, the mobile device consistently has (and therefore uses) fresh labels after the first five minutes. These labels will however be for routes which have been shortened due to lack of indicator-key pairs — the agent only starts to have enough indicator-key pairs at the end of the first five minutes, so that full-length labels will not be present until a little later.

Following this initial phase the agent and mobile device continue to acquire resources at their maximum rate. After about 900 seconds the agent passes the 750 indicator-key pairs mark and its rate of acquisition slows shortly afterwards at about 1100 seconds as the effects of the reduction of its rate of requesting indicator-key pairs take effect.

After 1500 seconds the mobile device has acquired all the labels it desires, and has pushed all the reused labels off the bottom of its stack; after that it will discard the initial short labels.

After 3200 seconds (53 minutes) the agent passes the 2000 indicator-key pairs threshold, and settles into its steady state. Whether at this point the remailers it is using will no longer be able to see the links between its indicator-key pairs and itself and each other will depend on what proportion of its indicator-key pairs are second-hand, i.e. have been obtained from the stock of a third remailer. This aspect has not been modelled.

9.4. Steady operation

During continuous operation the number of indicator-key pairs held by the remailer displays an oscillation around 2000. The sawtooth oscillations are due to the delay between the agent becoming aware that it wants or no longer wants more pairs and the results from its requests or lack of them returning to it.

As can be seen from the graphs the age of the most recent location data held by the agent — this is the data that will be used when responding to enquiries — is around 90-120 seconds. Thus the delay between the user moving and their agent being informed would be around two minutes.

This is three to four times the 30 second batching interval employed by the remailers. There is an artefact here, due to the fact that the message is passing many times through the same remailer: each time a message for remailing arrives it does so just after the queue has been sent, since it arrived as a result of that very queue run. One might expect, therefore, that in a system with several remailers the mean delay would be smaller.

The age of the address labels being used varies more widely, between 100 and 600 seconds; however, the original timestamp in an address label is not used for anything but checking that a label is moderately recent (less than ten minutes old, in the tests carried out) before using the location data in it. The fact that some labels arrive which are five or ten minutes old is therefore not a problem and does not represent a real lag in the system.

9.5. Effect of movements of mobile device

Figure A.3 (page 82) shows the effect if the mobile device moves to a brand new simulated location every 100 seconds. The system starts in its steady state, and the first movement occurs at 2000 seconds along the time axis.

The labels sent by the agent never manage to catch up with the mobile device; consequently it is as if the mobile device were unable to receive any transmissions; the effect of this is described below.

Later in the same graph (at 4400 seconds) the interval between movements was increased to 300 seconds (five minutes). In this case the system is able to operate normally, though a larger than usual number of indicator-key pairs is being used and the mobile device is using somewhat older address labels on average than usual.

In reality the user's location would not change exactly every 100 seconds, so that it is unlikely that the behaviour would be as pathological as is shown in the early part of figure A.3. Furthermore, the effects of the presence of only a single remailer on the round-trip time also have an effect here.

9.6. Effect of lost transmissions to and from mobile device

When half the broadcast packets between the mobile device and the transceivers are lost the startup time is greatly increased. The agent reaches its desired number of indicator-key pairs much more rapidly, because it is only supplying half as many to the mobile device. The mobile device takes much longer to reach the steady state, but it does reach it. See figures A.4 and A.5 (pages 83 and 84) — note that the horizontal scale on the full graph is compressed so that it shows the first 150 minutes, compared to only the first 100 in the earlier graph for startup without packet loss.

The mobile device's rate of acquisition of labels in the linear region from 1500s to 5500s is approximately 0.050 labels/second, about a tenth of its original value of 0.49 labels/second.

If the transmissions from the mobile device to the transceivers are entirely impossible the agent no longer receives claimed locations and so it no longer tries to transmit labels, whether or not it would be able to do so. The mobile device will not receive any fresh labels. This can be seen in Figure A.7 (page 86); the transmissions are cut off starting at 1000 seconds.

If transmissions in the reverse direction are impossible the agent receives claimed locations from the mobile device. Since the mobile device is not receiving any fresh labels the age of the labels it is using rises past the staleness timeout, beyond which the agent would no longer publish the information in the location announcements. The agent continues to try to send labels to the badge. This can be seen in Figure A.6 (page 85); again, the transmissions are cut off starting at 1000 seconds.

9.7. Summary

The test results show that a large number of messages cross the remailer network especially during startup. These messages will assist explicit padding in protecting against traffic flow analysis. With sufficient storage the mobile device is able to continue to function even during periods of very poor connectivity, at least well enough to re-establish normal operation reasonably quickly when the situation improves. High rates of message loss can be sustained without serious problems.

Chapter 10. Query handling user interface

Now that we have described and developed the core system, this chapter will outline the requirements for the portion of the system which implements the user's access control policy for their location information, and for the interface they use to specify that policy.

One possible and straightforward solution to these requirements is presented.

10.1. Framework

The core agent program refers queries about the user's location to a separate program provided by the user; this separates the user's policy from the core infrastructure parts of the agent which handle tracking the mobile device, gathering and issuing address labels and so forth.

Most users will not have the skill and inclination to write such a program themselves; therefore an easily customisable program must be provided.

Ideally the user would be able to impose arbitrarily complicated conditions on the release of information about their location. This can be supported if the customisation is a full programming language. This language shoud be reasonably simple for the user to program, and not hard to implement.

John Ousterhout's Tool Command Language [Ousterhout93] provides a good basis for solving this problem; it is a small easily-embeddable language with a fairly straightforward syntax. With the addition of a few primitives allowing the user's script to find the properties of a request and specify the response it will serve for our purpose.

The 'event loop' for reading incoming requests and processing them should not need to be implemented by the user; instead, the user's specification should be run once for every incoming query.

10.2. Details of policy facilities

Information about the incoming request for location information is made available through an additional Tcl command `query-info`. It takes one argument, specifying what information it should return:

`type`
: `person` or `location`, depending whether the enquirer wishes to know the current location of a user, or information about users at a particular location; the core program will not pass to the decision program queries for locations other than the one where the user is.

`sign-response`
: Whether the enquirer would like the response certified; a string `no`, `user` or `transceiver`, to indicate that no signing is required, or that the enquirer would like the user to sign the response themselves or to acquire an anonymous certificate of location and forward it.

`enquirer`
: The identity of the enquirer (if known); this is a list whose first element specifies the type and the rest of which identifies a public key or a network address.

`enquirer-location`
: The location of the enquirer, if they chose to obtain a location certificate and send it with the query.

Information about the real current location of the user and an estimate of how accurate and recent this is is available through another additional Tcl command, `mobile-info`. It also takes one argument, specifying what information it should return:

`location`
: The user's most recent sighting.

`age`
: The age of the user's most recent sighting, in seconds, determined by comparing the transceiver's timestamp on the sighting with the current time as seen by the agent.

`confidence`
: An estimate of the proportion of recent messages from the mobile device which were received by the agent. This is an alpha-smoothed mean of the density of recent position announcements received by the agent, as an integer in units where the mean transmission frequency of the badge is 100. Note that this number may exceed 100.

The user's specification, a Tcl script, uses this command and any other Tcl primitives it wishes and then indicates its decision using the `query-reply` command, with one of the following sets of arguments:

`accept`
: Supply the requested information to the enquirer.

`accept sign`
: Supply the requested information to the enquirer, and sign it.

`accept send-certificate`
: Obtain an anonymous certificate of location and forward it to the enquirer.

`reject`
: Do not supply the information to the enquirer. In the case of a query about a location the core program will not reply at all.

`reject where`
: Do not supply the information to the enquirer, but ask them to resubmit the query with a certificate of location. This only makes sense in the case of a query about a person, since any response to a query about a location requesting more information would identify the responder.

`reject who`
: Do not supply the information to the enquirer, but ask them to resubmit the query signed with their public key, to identify them. Again, this only makes sense in the case of a query about a person.

A command is provided for limiting the rate at which queries are accepted:

`rate-limit` *mean-delay half-life identification*
: This command maintains a set of alpha-smoothed query counters. Each different set of identification arguments has its own counter. When the command is executed a check is made to see whether an event now would bring the mean interval between requests (averaged approximately over the past half-life) under the mean-delay specified. If so the command returns 0 and no event is recorded, otherwise it returns 1 and an event is recorded.

A command is provided for limiting the information available in response to any accepted query, and another for lying. Each takes as a first argument an *information-name*, which is one of the acceptable arguments to `mobile-info`.

`response-hide` *information-name*
: This command ensures that the piece of information in question will not be sent if `query-reply` is used to specify that the query should be answered.

`response-tell` *information-name*
: This command ensures that the piece of information in question will be sent in any reply to the query. This is the default.

`response-lie` *information-name fake-value*
: This command causes any reply to the query to contain *information-name fake-value* as the value for the piece of information in question. The use of this with `query-reply accept sign` is ill-advised, and the use with `query-reply accept certificate` is not possible and will be treated as an error.

Additionally, sometimes it will be useful to include random responses. A cryptographically strong random number generator is available through `crypto-random-sample`, with one or more arguments. Each call produces a new value.

`boolean` *probability*
: 1 with probability *probability*, 0 otherwise.

`discrete-uniform` *min max*
: A sample from the discrete uniform distribution of integers not less than *min* and not greater than *max*.

`continuous-uniform` *min max*
: The continuous uniform distribution of real numbers not less than *min* and strictly less than *max*.

`normal` *mean variance*
: The normal distribution with the specified mean and variance.

`exponential` *mean*
: The exponential distribution with the specified mean.

`poisson` *mean*
: The Possion distribution with the specified mean.

Tcl's `return` command can be used to stop the parsing of the script immediately and return the answer as specified by the most recently read settings; otherwise the interpreter will read the whole script with later settings overriding earlier ones.

After the response to each query is determined the agent will automatically log the information it had about the query and the response which was sent.

The facility to cause inaccurate responses to be sent exists in the underlying protocols, and its inclusion in the standard command set will hopefully make the possibility obvious and encourage the users of location information to obtain proper certificates of location or at least a signature from the user before relying on this information.

From the user's point of view it would probably be considered a feature.

10.3. Example

```
if {[string match "*tearoom*" [mobile-info location]]} {
  response-lie location "Austin 4 corridor"
  response-lie confidence [expr {[mobile-info confidence]/2}]
  response-lie age [expr {[mobile-info age]+20}]
  set hiding 1
} elseif {[string match "*Eagle public house*"
          [mobile-info location]]} {
  response-lie location "Tower Ground floor corridor"
  response-lie confidence [crypto-random-sample possion 10]
  response-lie age [crypto-random-sample exponential 900]
  set hiding 1
} else {
  set hiding 0
}
if {[query-info sign-response] != no &&
    [rate-limit 600 3600 sign] && !$hiding} {
  if {[string length [query-info enquirer]] == 0} {
    query-reply reject who
  } elseif {[query-info enquirer] == {local-user diederik}} {
    query-reply accept send-certificate
  } else {
    query-reply reject
  }
  return
}
if {![string length [query-info enquirer]]} {
  query-reply reject who
} elseif {![string length [query-info enquirer-location]]} {
  query-reply reject where
} elseif {[rate-limit 120 1800 identified]} {
  query-reply accept
}
```

This user has a moderately but not excessively complicated policy. They generally wish only to be seen by people who reveal their own identity and location, and even then at most once every two minutes in total for all enquirers. When requested to certify their own location they will only do so if the enquirer is the local user ‘diederik’, and then at most once every ten minutes.

As an added complication they are shy about the amount of time they spend in the tearoom and in the pub; if they are there then they want people to be told that they’re wandering around the building or have just left, respectively. If they are asked to sign such a response they will refuse.

10.4. Limitations and weaknesses

In principle the interface presented here places no limits on the behaviour which users can program into their agents. However, it does have some weaknesses.

If the user wishes to implement a policy based on relationships between queries other than by using a simple rate-limiter they must implement this entirely themselves by using global, long-term Tcl variables. If this state is to be preserved if and when the agent, and consequently the decision program containing the Tcl interpreter, die and are restarted (as may happen from time to time, depending on the reliability of the underlying system hosting the agent and of the agent and associated code) then the user must make their own arrangements for storing the information in a permanent place such as the filesystem of the system hosting their agent.

Likewise, there is no facility defined for analysis of the logfiles created; the user must handle this themselves.

The use and distribution of location information presents aggregation problems which are ill-understood, and in any case, the user's policy may reveal information the user wants to keep secret just by whether an enquiry is accepted or rejected. The interface presented gives the user no assistance in avoiding even the most obvious pitfalls.

For some users even writing a simple script will be very difficult; many end-users of modern computing systems are more used to graphical user interfaces than command lines and files containing lists of instructions in what may seem an arcane syntax. If so then the arrangements described above could at best serve as the implementation substrate for a graphically based agent configuration manager. If such a graphical interface were to be designed care would have to be taken that the possible logical structures of the rules for accepting, rejecting and logging queries are sufficiently powerful.

It may of course be possible to provide extra information in requests or return it in queries. This at least is not a serious problem as the scheme described above is easily extended.

Chapter 11. Attacks and countermeasures

This chapter will discuss some of the attacks possible against the system described earlier, in various environments, and what countermeasures might be taken against them if they are thought to be a serious risk.

11.1. Subverting the mobile device

The mobile device is a trusted component of the system, and if it is compromised (for example, by causing it to transmit its identity unencrypted as well as taking part in the protocol as designed) the security of the system as a whole is compromised for that user.

How difficult it is to introduce rogue programming into the mobile device, and how likely this is to go undetected, will depend on the environment. In some environments the mobile devices will rarely be handled by anyone except the user, and the software could be loaded into them (or their ROMs programmed) by people trusted by the user (for example, their union representatives).

In other environments the mobile device will frequently be left unattended and may need to be returned often for servicing (repair, replacement of batteries, etc.). In this case it may be desirable to have some kind of independent checking of the signals coming out of the mobile device, or to make the device user-serviceable so that they can bring it to a trusted party for checking.

In any case a level of checking could ensure that widespread abuse would run a real risk of being discovered, and the likely social impact of such a discovery would discourage the abuse.

11.2. Traffic flow analysis

The system relies for its security on the premise that there are many mobile devices and agents in the system, so that the traffic between any pair is not easily traced.

When the number of users becomes very small this assumption ceases to hold. This can be partially offset by reducing the number of different remailers used by the different users and by generating extra cover traffic in the remailers and agents to hide the real traffic.

However, if it is known that there is only one user in a particular building then any scheme of cover traffic generation would have to generate real transmissions of messages at many other locations besides the one where the user really is. This would require a parallel network of trusted hardware to generate this cover traffic, and once such a trusted network is postulated the necessity for the use of remailers to hide the users' locations from the network vanishes.

Ultimately, though, the mistrust which might motivate the use of a system such as that described here might not exist in the small groups where traffic flow analysis is a problem.

[babel96] contains an enumeration of types of attacks on remailers, many of which are based on traffic flow analysis. Content correlation at remailers is avoided by encryption, as in Gülcü and Tsudik's scheme; size correlation is made more difficult by the regular nature of the messages passing through the system, but must ultimately be avoided by padding. Time correlation is avoided by batching very similar to their interval batching. Active attacks based on filling batches with dummy messages are prevented because interval batching is used; message replay attacks are avoided by making the number of transmissions of an output message independent of the number of times the corresponding input message is received.

11.3. Subverting the user's agent

The user's agent is the other piece of the ultimately-trusted computing base. If it is compromised then the attacker can track the user using the information which the user's agent needs to be able to maintain communication with the mobile device.

How easily subverted the agent is will depend strongly on the political and organisational arrangements. If the agent runs on a dedicated computer close to the user which untrusted people have little or no administrative control over it will be hard to subvert.

At the other extreme it might run together with many other users' agents on a single fairly insecure timesharing system. In this case successfully attacking just that one system will breach the security of all the agents affected. However, continued monitoring of the users' locations would probably require a continuing attack on the system and those processes, which would risk discovery. Furthermore, in many such arrangements an attack on the user's agent might well automatically involve an attack on a system which is used for other purposes, so that breaches of its security might be taken more seriously than otherwise. This might help deter such attacks.

In any case, many users of electronic mail systems running on fairly vulnerable timesharing systems seem happy (justifiably or not) to run the risk that their personal or sensitive messages might be discovered by an intruder.

11.4. Subverting remailer(s)

If many remailers are used then all or most of them must be compromised before the security of the system as a whole is damaged.

As with the agents, subverting all or most of them could be made very difficult if they are under independent administrative control and run on separate computers. However, in extreme cases they might all run on the same fairly insecure timesharing system, giving only marginally better security than running just one single remailer.

The issues and tradeoffs here are very similar to those with agents; indeed, in the implemented system the remailers and agents are the same program, and it would be likely that colleagues' and friends' agents will be acting as remailers for each other.

11.5. Design and implementation errors

It is possible that there are errors in the implementation which could lead to a security failure. The likely number and severity of such errors is hard to estimate at this stage; more information may become available if the system is more widely fielded.

Many protocols proposed in the past have had problems, some of which have been found only years afterwards. These problems have sometimes led to a complete breakdown of the security of fielded systems, but usually the fielded systems have been able to be fixed before this occurred. Formal verification tools, which are intented to help detect and prevent protocol failure, have not yielded any very useful results when applied to the protocols in use here.

The requirement for strong anonymity and the constraint on the size of messages sent to and from the mobile device have made it difficult to include all the information in messages that one might like to include to satisfy general robustness principles [AN96]. However, where possible the identities of the entities involved are included, and nonces are used to tie information together across messages where the identity of sender is not to be known to the recipient.

11.6. Cryptanalysis

The use of widely-known algorithms in straightforward ways will hopefully make this an unrealistic threat if keys are properly generated and other aspects of the implementation are sound.

DES is the perhaps the world's most-studied block cipher, and its chief weakness is now thought to be its short key length. The triple-DES variant used is CBC using a triple-DES encipher-decipher-encipher as the component block cipher; feedback is taken from the final output of each block and used for the initial input to the next block, rather than feeding back into the intervening inputs of the next block (which would be equivalent to three CBC's composed, rather than a CBC made from three DES operations). This avoids recently-discovered attacks on DES multiple modes.

Messages are not merely encrypted using triple-DES CBC; each symmetric key encryption includes padding and a MAC.

RSA is a well-understood, popular, well-studied and straightforward public-key cryptosystem. It is used in a well-understood and standard manner, and only for confidentiality (not for signature). The use of an intermediate per-message symmetric key prevents chosen-ciphertext attacks on RSA, and other similar attacks [PP89].

11.7. Best attack

It seems likely that in environments where users' agents and remailers are running on multipurpose multiprocessing systems the easiest route for an attack would be to compromise the security of those systems using weaknesses in other programs running on them. Depending on the number and nature of flaws in the implementation these may yield a good avenue for attack.

Attacks based on modified mobile device software are likely only to become relevant when the users' agents and remailers run on fairly dedicated and secure systems.

More sophisticated attacks based on traffic flow analysis, protocol failure or cryptanalysis are not likely to be of practical use to an attacker, but may provide interesting material for further discussion and research.

In systems with only a small number of users, traffic flow analysis, based on correlation of messages from the same users' mobile device, will be more feasible.

Chapter 12. Conclusions and directions for future work

A mechanism has been presented here which allows a user who is the subject of location tracking technology to effectively control to whom this information is available and how, even when they distrust a large proportion of the computing infrastructure which implements the location tracking system. They are not required to trust the network of 'base stations' with which the mobile device communicates.

The mechanism is based initially on the 'mixes' proposed by David Chaum in [Chaum81] but has been enhanced to eliminate the use of public key cryptography in the mobile device, thus avoiding the need for the transmission of large messages to the mobile device and the need for the mobile device to be a powerful computer.

An abstract description of the cryptographic protocols involved is given, with details of the core functions for location tracking and also those which support these functions of the system. Protocols and mechanisms for implementing functionality other than simple location tracking are discussed and, for certain other functionality, presented.

A sample implementation of the core of this mechanism for a particular location-tracking system, ORL's Active Badges, has been described and tested. Its performance under various simulated conditions has been measured and found to be adequate.

The strengths and weaknesses of the design are discussed. The most likely failure mode in most environments is an integrity failure of the fixed-location trusted computing systems on which the software runs; if these are moderately secure then the next best attack is on the integrity and accuracy of the software on the mobile device which the user carries to enable the location tracking, whose correct operation the user will usually have only a limited ability to verify. In environments with a small number of users it becomes more difficult to hide any one user amongst the others.

A description is given of a small user interface with which users can express the security policy they wish to be applied to the information about their location. Users write their policy in a moderately straightforward and easily extensible textual language permitting simple or complex rules. This user interface language is directly adequate for most simple policies and can be put to use for complex ones.

It is hoped that technology such as that presented here will encourage the more widespread acceptance of location tracking systems amongst organisations and sceptical individuals than has been the case until now, and that mechanisms like the one presented here or others achieving similar goals will encourage operators of existing systems to give their subjects more control over the way the information about them is used.

A formal logic was designed with a view to analysing the protocols used, existing tools having been found inadequate. This logic also has some difficulty verifying the protocols in question because of the lack of a means of expressing the requirement that the relationship between two things be kept confidential. However, the logic is useful in verifying more conventional protocols and more powerful than traditional logics of its kind. It is hoped that this logic will help designers of new protocols avoid errors, and to suggest to designers of logics a way to improve the power of their creations.

Similar problems occur in other mobile networking and communications fields. For example, mobile telephones [Rahnema93] [cdpd93] and some pagers have similar problems with location tracking, and the use of technology similar to that presented here could usefully be investigated. Other location-related application areas which might benefit from similar treatment are alarm and tracking systems for expensive equipment and items; possibly even road toll payment systems should be investigated.

Further interesting research might be done on the statistics and probability theory of the amount of information that remailers gain through knowledge of linkages between the various indicator-key pairs that they see.

The properties required of a graphical user interface for control over access to location information, suitable for users with little computing experience, should be investigated.

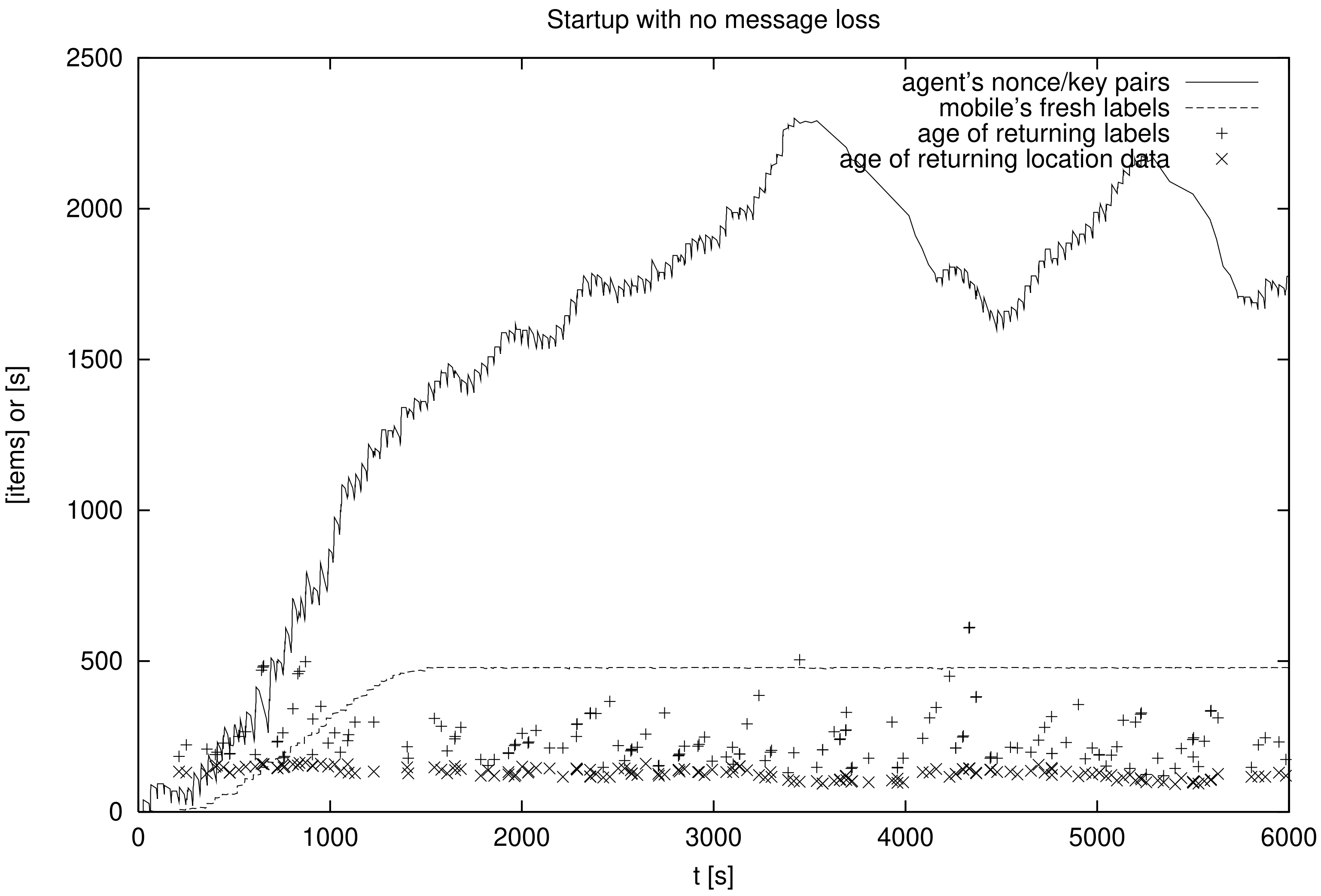


Figure A.1. Startup with no message loss

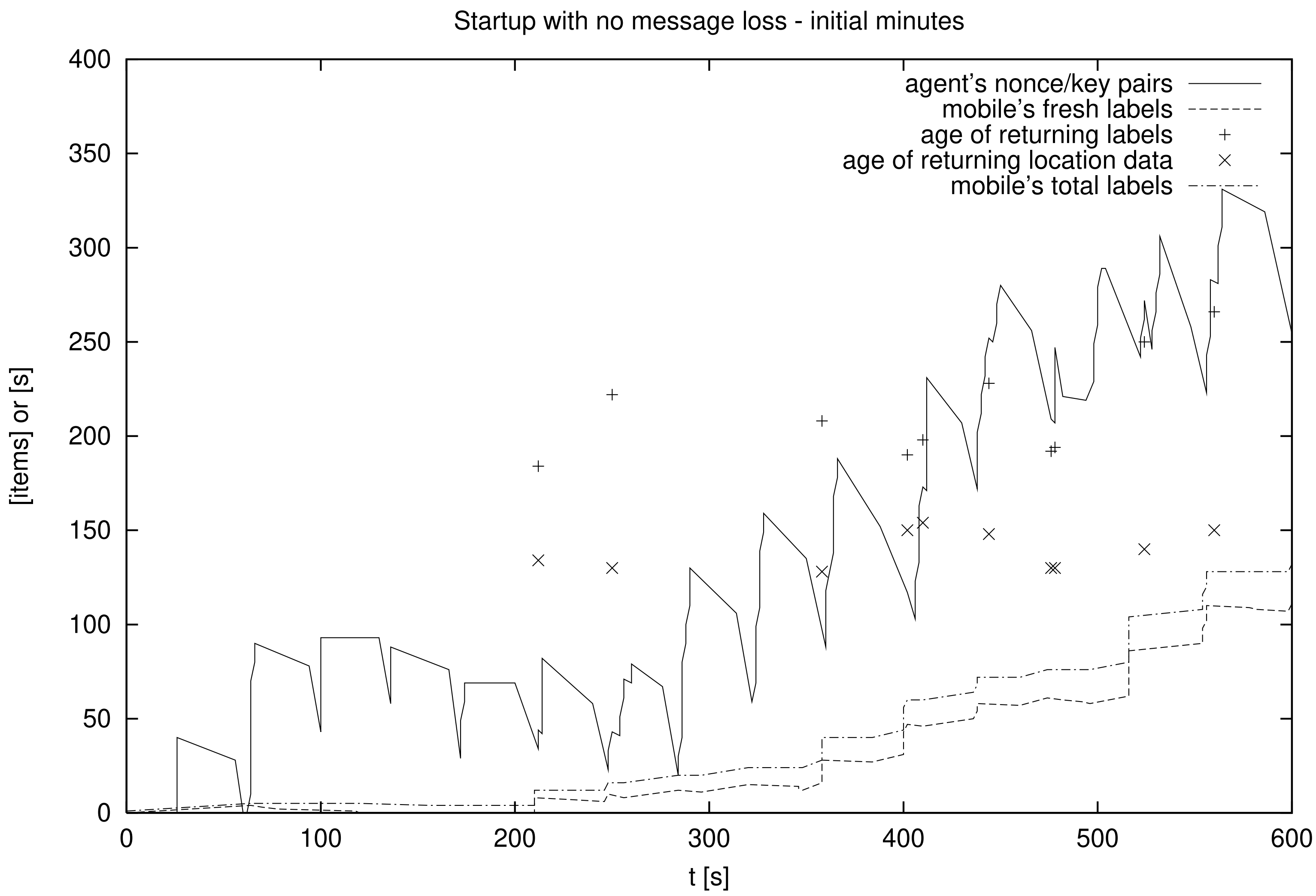


Figure A.2. Startup with no message loss — initial minutes

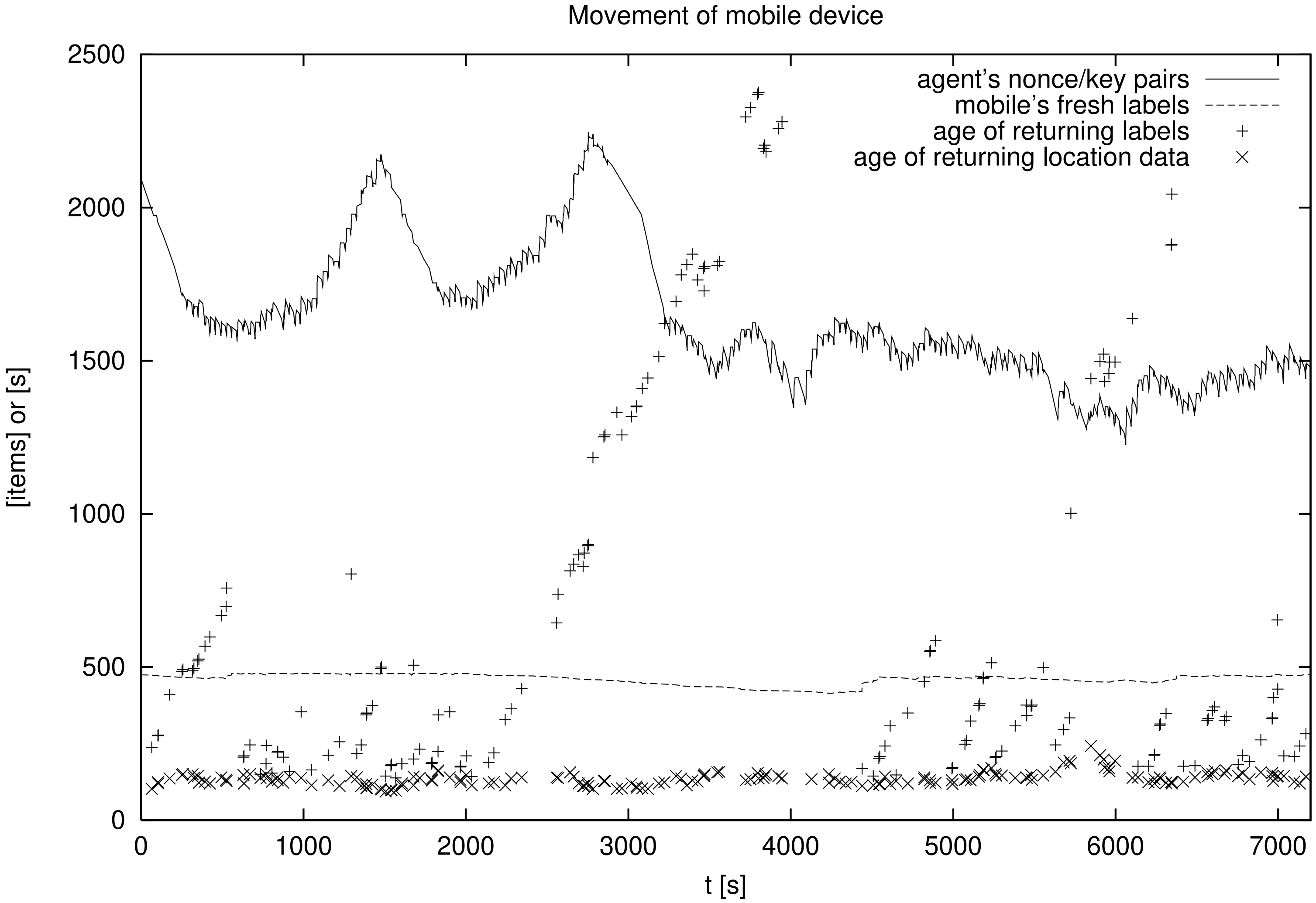


Figure A.3. Movement of mobile device

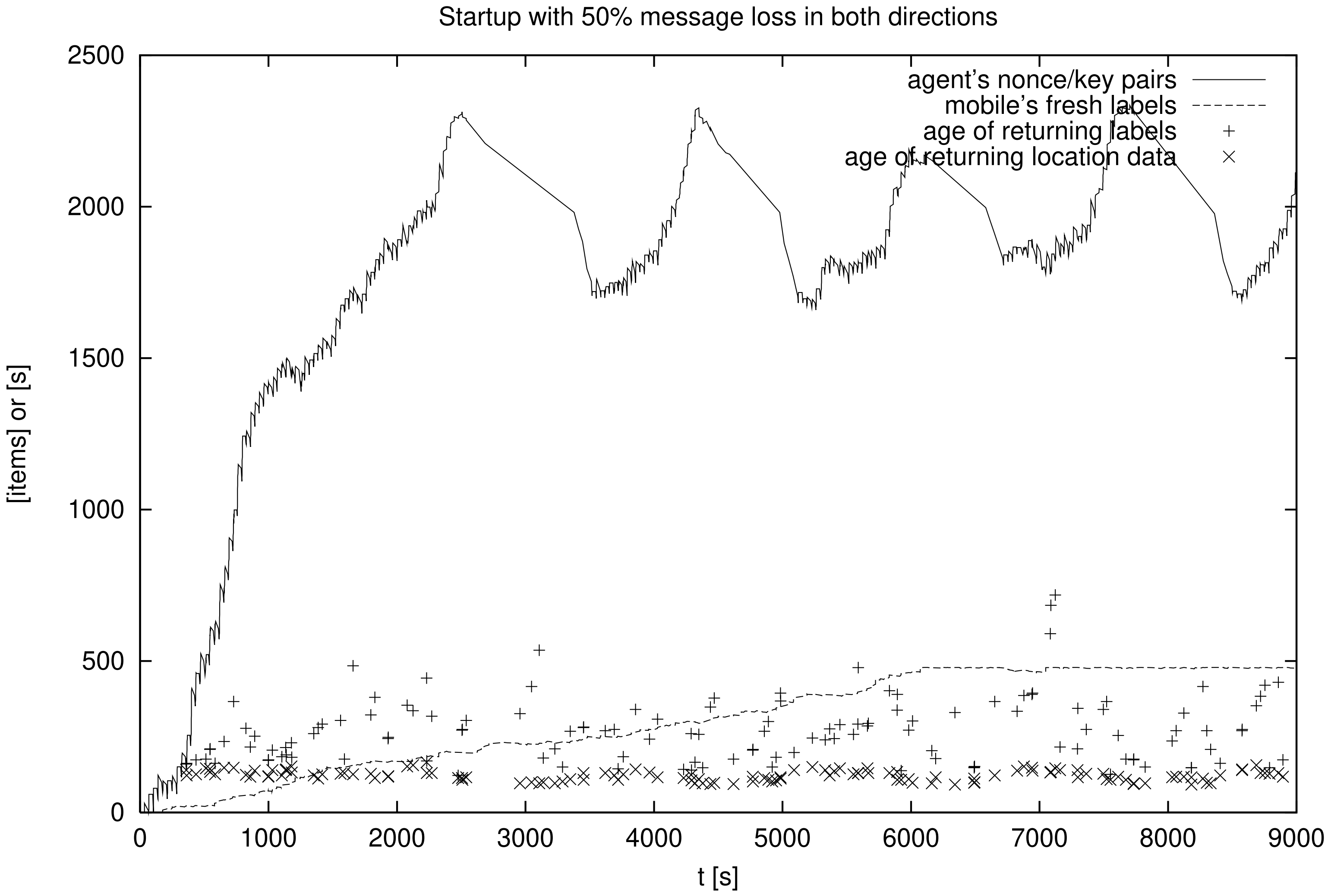


Figure A.4. Startup with 50% message loss in both directions

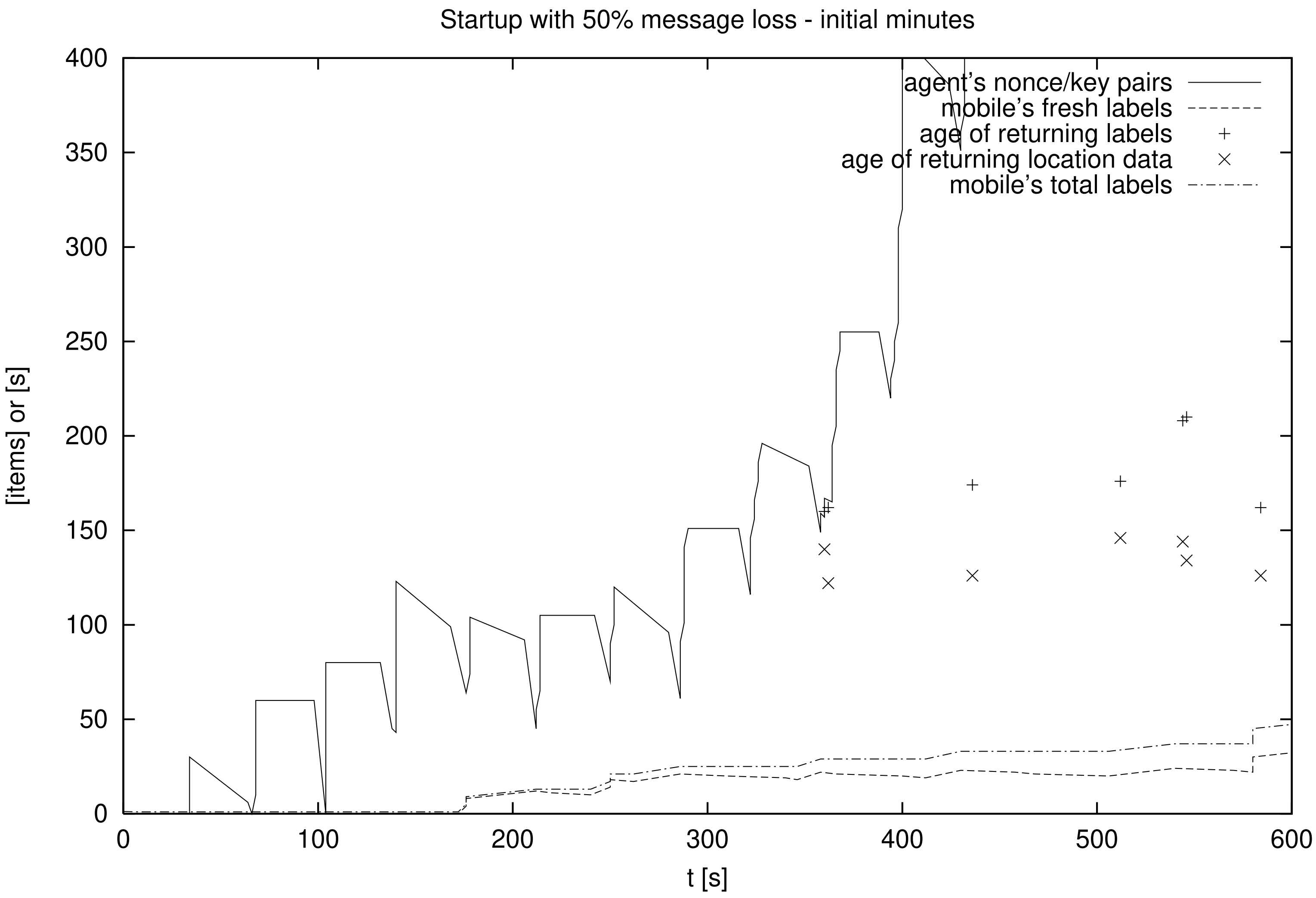


Figure A.5. Startup with 50% message loss — initial minutes

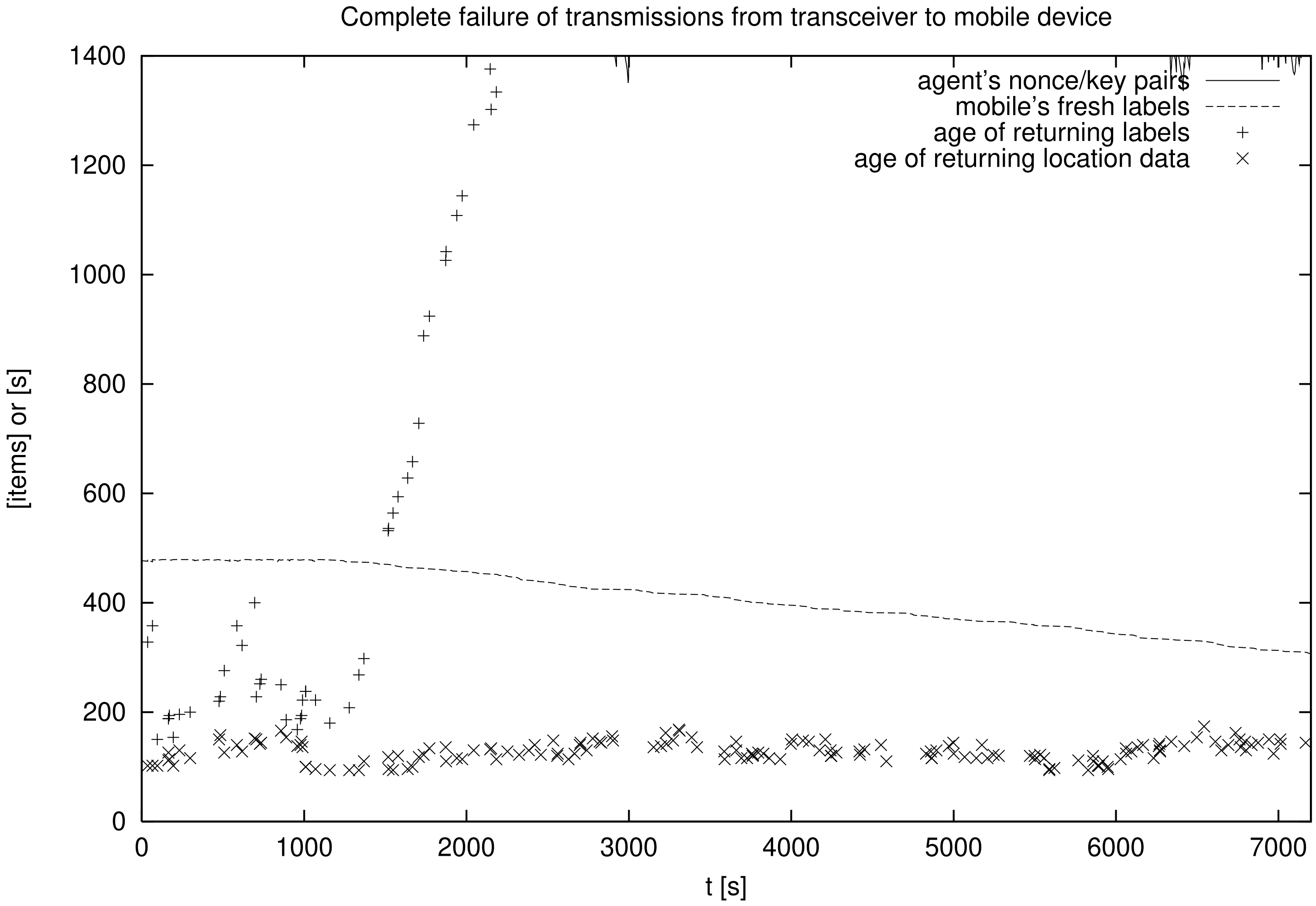


Figure A.6. Complete failure of transmissions from transceiver to mobile device

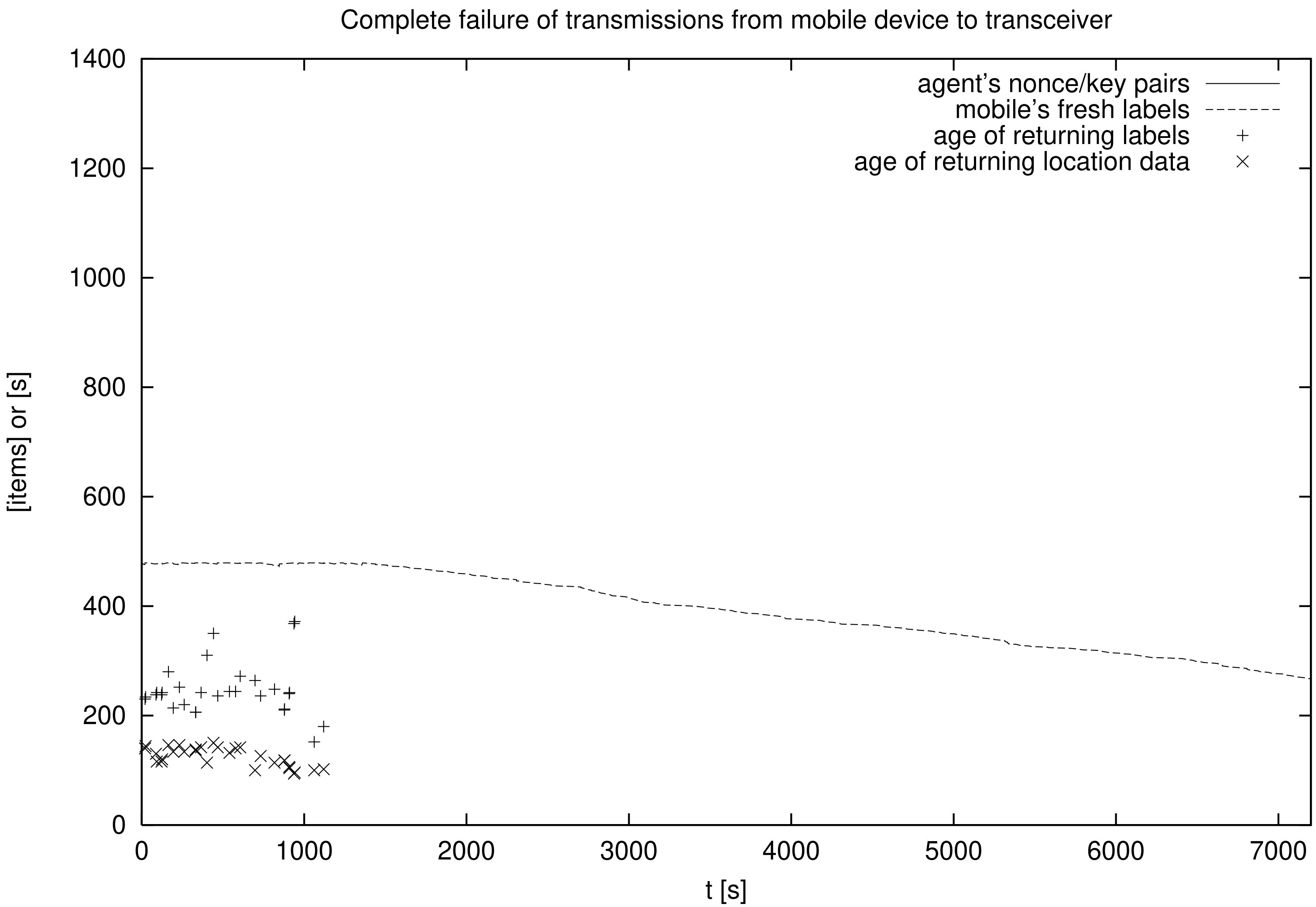


Figure A.7. Complete failure of transmissions from mobile device to transceiver

Appendix B. Expanded examples of protocols

This appendix contains some example protocol runs' messages with all their nested contents expanded.

In these examples there is one mobile device, *B*, with its agent *A*, a transciever *S* and two remailers *P* and *Q*.

B.1. Protocol with public-key done by mobile device

This is the protocol described in section 6.2 (page 25 onwards).

The mobile device constructs an address label for itself, by recursively encapsulating its own name with the addresses for the entities the message will pass through, in reverse order.

Firstly, E_A is just *B*, the name of the mobile device.

Then E_Q is:

$$\{N_{B,Q}\}_{K_{AB}}, \{f(N_{B,Q})\}_{K_{AB}}, A, \{E_A\}_{K_A}$$

or

$$\{N_{B,Q}\}_{K_{AB}}, \{f(N_{B,Q})\}_{K_{AB}}, A, \{B\}_{K_A}$$

($N_{B,Q}$ is generated by B; A and B start off sharing K_{AB}.)

Now E_P is:

$$\{N_{B,P}\}_{K_{AB}}, \{f(N_{B,P})\}_{K_{AB}}, Q, \{E_Q\}_{K_Q}$$

and E_S is:

$$\{N_{B,S}\}_{K_{AB}}, \{f(N_{B,S})\}_{K_{AB}}, P, \{E_P\}_{K_P}$$

So the entire address label transmitted by the mobile device to the transceiver via the mobile link is:

$$\begin{gathered}\{N_{B,S}\}_{K_{AB}}, \{f(N_{B,S})\}_{K_{AB}}, P, \\ \{\{N_{B,P}\}_{K_{AB}}, \{f(N_{B,P})\}_{K_{AB}}, Q, \{\{N_{B,Q}\}_{K_{AB}}, \{f(N_{B,Q})\}_{K_{AB}}, A, \{B\}_{K_A}\}_{K_Q}\}_{K_P}\end{gathered}$$

The transceiver, which knows the location *L*, appends the encrypted location (and the keying information for *A*) and forwards to *P*:

$$\{\{N_{B,P}\}_{K_{AB}}, \{f\,(N_{B,P})\}_{K_{AB}}, Q\,,\ \{\{N_{B,Q}\}_{K_{AB}}, \{f\,(N_{B,Q})\}_{K_{AB}}, A\,,\ \{B\}_{K_A}\}_{K_Q}\}_{K_P},$$
$$\{N_{B,S}\}_{K_{AB}}, \{L\}_{\{f\,(N_{B,S})\}_{K_{AB}}}$$

P decrypts the first half and superencrypts the second, forwarding to *Q*:

$$\{\{N_{B,Q}\}_{K_{AB}}, \{f\,(N_{B,Q})\}_{K_{AB}}, A\,,\ \{B\}_{K_A}\}_{K_Q}, \{N_{B,P}\}_{K_{AB}}, \{\{N_{B,S}\}_{K_{AB}}, \{L\}_{\{f\,(N_{B,S})\}_{K_{AB}}}\}_{\{f\,(N_{B,P})\}_{K_{AB}}}$$

Q does the same, forwarding to *A*:

$$\{B\}_{K_A}, \{N_{B,Q}\}_{K_{AB}}, \{\{N_{B,P}\}_{K_{AB}}, \{\{N_{B,S}\}_{K_{AB}}, \{L\}_{\{f\,(N_{B,S})\}_{K_{AB}}}\}_{\{f\,(N_{B,P})\}_{K_{AB}}}\}_{\{f\,(N_{B,Q})\}_{K_{AB}}}$$

Now *A* can use K_A^{-1} to recover *B*'s identity, and K_{AB} to recover all the $N_{B,X}$'s and so find *L*.

B.2. Protocol with no public-key in mobile device

This section is an example of the protocol described in section 6.3 (page 26 onwards); examples of the acquisition of indicator-key pairs to make address lables and the delivery of the labels to the mobile device follow a little later.

The address labels are constructed by the agent, starting with its own details. So, E_A is:

$$N_A\,,\ B\,,\ T_A$$

(*B* is the identity of the mobile device and T_A is a timestamp generated by the agent when the label is constructed.)

The agent encapsulates this using pairs from *Q*, and some pairs it makes up for itself. E_Q is

$$I_{Ai}\,, K_{Ai}\,, A\,, I_{Al}\,, \{E_A\}_{K_{Al}}$$

E_P is

$$I_{Aj}\,, K_{Aj}\,, Q\,, I_{Qm}\,, \{E_Q\}_{K_{Qm}}$$

and E_S is

$$I_{Ak}\,, K_{Ak}\,, P\,, I_{Pm}\,, \{E_P\}_{K_{Pn}}$$

So, when rendered in full the whole address label and nonce (N_A , which is transferred to the mobile device with the labels) sent by the mobile device to the transceiver, is:

$$N_A , I_{Ak} , K_{Ak} , P , I_{Pn} , \{ I_{Aj} , K_{Aj} , Q , I_{Qm} , \{ I_{Ai} , K_{Ai} , A , I_{Al} , \{ N_A , B , T_A \}_{K_{Al}} \}_{K_{Qm}} \}_{K_{Pn}}$$

The transceiver adds the location data L and the nonce N_A , encrypted using I_{Ak} , and forwards the result to P as follows:

$$\boxed{I_{Pn} , \{ I_{Aj} , K_{Aj} , Q , I_{Qm} , \{ I_{Ai} , K_{Ai} , A , I_{Al} , \{ N_A , B , T_A \}_{K_{Al}} \}_{K_{Qm}} \}_{K_{Pn}}} ,$$
$$\boxed{I_{Ak} , \{ L , N_A \}_{K_{Ak}}}$$

P decrypts E_P , reencrypts the data portion, and forwards to Q:

$$\boxed{I_{Qm} , \{ I_{Ai} , K_{Ai} , A , I_{Al} , \{ N_A , B , T_A \}_{K_{Al}} \}_{K_{Qm}}} , \boxed{I_{Aj} , \{ I_{Ak} , \{ L , N_A \}_{K_{Ak}} \}_{K_{Aj}}}$$

Q does likewise, sending to A:

$$\boxed{I_{Al} , \{ N_A , B , T_A \}_{K_{Al}}} , \boxed{I_{Ai} , \{ I_{Aj} , \{ I_{Ak} , \{ L , N_A \}_{K_{Ak}} \}_{K_{Aj}} \}_{K_{Ai}}}$$

Now A can use K_{AB} to recover the K_{Ax} for each I_{Ax} , strip off the layers of encryption, so that it now has:

$$\boxed{N_A , B , T_A} , \boxed{L , N_A}$$

A compares the two copies of N_A , checks the timestamp, and if all is well it records the mobile device B as having location L.

B.3. Acquisition of indicator-key pairs

This section is an example of the protocols described in sections 6.4 and 6.8.

Let us suppose that the agent A already has some indicator-key pairs for the remailer it wishes to be nearest to it in the chain, Q, and now needs some for the next remailer, P.

It generates the request for z pairs:

$$P,\ z,\ N_A$$

This needs to be attached to its return information via Q, which is constructed as above using its existing pairs for Q:

$$\boxed{P,\ z,\ N_A},\ \boxed{I_{Aj}\,,K_{Aj}\,,Q\,,I_{Qm}\,,\{I_{Ai}\,,K_{Ai}\,,A\,,I_{Al}\,,\{P,\ z,\ N_A\}_{K_{Al}}\}_{K_{Qm}}}$$

It wishes P to receive this message indirectly, via Q, so it sends to Q:

$$\{P\,,\{\ \boxed{P,\ z,\ N_A},\ \boxed{I_{Aj}\,,K_{Aj}\,,Q\,,I_{Qm}\,,\{I_{Ai}\,,K_{Ai}\,,A\,,I_{Al}\,,\{P,\ z,\ N_A\}_{K_{Al}}\}_{K_{Qm}}}\ \}_{K_P}\}_{K_Q}$$

Q decrypts this using $K_Q{}^{-1}$, sees P's name and some data, and forwards to P:

$$\{\ \boxed{P,\ z,\ N_A},\ \boxed{I_{Aj}\,,K_{Aj}\,,Q\,,I_{Qm}\,,\{I_{Ai}\,,K_{Ai}\,,A\,,I_{Al}\,,\{P,\ z,\ N_A\}_{K_{Al}}\}_{K_{Qm}}}\ \}_{K_P}$$

P decrypts this using $K_P{}^{-1}$. The first portion is a request for indicator-key pairs, and the second portion is the anonymous return address. It generates up to z pairs and forwards them using the anonymous address, sending via Q:

$$\boxed{I_{Qm}\,,\{I_{Ai}\,,K_{Ai}\,,A\,,I_{Al}\,,\{P,\ z,\ N_A\}_{K_{Al}}\}_{K_{Qm}}},$$

$$\boxed{I_{Aj}\,,\{y\,,N_A\,,I_{P(k+1)}\,,K_{P(k+1)}\,,\ \ldots,\ I_{P(k+y)}\,,K_{P(k+y)}\ \}_{K_{Aj}}}$$

To Q this looks just like a normal forward-path remailing; it unpacks the first (address) portion and reencapsulates the second, forwarding to A:

$$\boxed{I_{Ai}\,,K_{Ai}\,,A\,,I_{Al}\,,\{P,\ z,\ N_A\}_{K_{Al}}},$$

$$\boxed{I_{Ai}\,,\{I_{Aj}\,,\{y\,,N_A\,,I_{P(k+1)}\,,K_{P(k+1)}\,,\ \ldots,\ I_{P(k+y)}\,,K_{P(k+y)}\ \}_{K_{Aj}}\}_{K_{Ai}}}$$

Now A strips off the layers of encryption, so that it has:

$$\boxed{P,\ z,\ N_A},\ \boxed{y\,,N_A\,,I_{P(k+1)}\,,K_{P(k+1)}\,,\ \ldots,\ I_{P(k+y)}\,,K_{P(k+y)}}$$

It checks that y is no more than z, and that the two copies of N_A match, and then records the I_{Px}, K_{Px} pairs as belonging to P.

Appendix C. Corrections from first version

This appendix summarises the changes made between the first (submitted) version and this revised version.

In sections 5.1 and 5.4 the relationship between cover traffic and remailers/mixes is more fully explored. The fact that the user's agent is in a good position to organise the generation of cover traffic is noted more clearly in section 6.10. In section 5.2 the nature of the sophistication in the Dining Cryptographers scheme [Chaum88] compared to plain cover traffic (namely, the unconditional security) is noted.

There is a new bibliography entry [babel96] which is discussed in sections 3.2.1 and 11.2.

Section 7.2 now starts with a general overview of the structure of the logic, and section 7.2.3 explains the meaning of the notation of the logic's rules.

There is a new section 7.2.1 which introduces the types of expressions in the protocol analysis logic, and the meanings of those types. Notation is now specified which distinguishes those types, and this is now used throughout chapter 7. Section 7.2.2 states the expected types of the operands to operators, and the types of the resulting expressions, and has been clarified.

Section 7.3 is now clearer about when apparently redundant information from the idealised messages may be left out of the actual transmitted representations of those messages.

The **believes** symbol in the logic has been changed to **says**, and the explanation in section 7.2.2.2 changed to clarify the symbol's use in preventing reflection and related attacks.

The definition of the $\models$ symbol in section 7.2.2.2 has been changed from a statement of belief to a statement about message origin.

There is a new section 7.8.1 which discusses the requirement by the logic for the sender's identity to be present in protocol messages, as it affects potential analysis of the remailer protocols, and offers two possible solutions.

Two more example protocols, public-key Needham-Schroeder (from [NS78], a new bibliography entry) and the wide-mouthed frog, are considered fairly briefly in sections 7.6 and 7.7. With respect to public-key Needham-Schroeder, Gavin Lowe's attack [Lowe95] is mentioned.

Section 6.10 now discusses the relationship between the security provided and lengths of chains of remailers, the size of the network and the level of traffic at each remailer, as well as considering how the provision of remailers might be organised.

There is a new appendix B, containing expanded examples of the protocol messages from chapter 6. For consistency, some of the notation in chapter 6 have been adjusted to agree more closely with that required in the appendix to show a number of levels of the protocol at once.

The recursive nature of E_P, E_Q etc. in chapter 6 is more fully explained, and the notation changed from the previous *stuff for Q*. The enquirer in section 6.9 has been renamed U (previously E) to avoid confusion.

The {key indicator, key} pairs introduced in section 6.3 have been renamed (from nonce/key pairs), and the notation and terminology adjusted accordinly throughout.

There is a new section 8.1 which describes the constraints on Active Badge hardware. It contrasts the capabilities of the fielded Badge system with those of which would be provided were the system to be designed. It notes that due to this discrepancy a laptop computer was used to simulate the badge during the testing.

In section 3.1, the full definition of the matrix C_5 from [SMA95] is given.

The paragraph which considers operators with unethical intent (page 13) has been clarified.

The style of the bibliography was made more uniform (authors are now always named only by their initials). Certain typographical and formatting errors have been corrected, including a missing author name in the bibliography, some key indicators which were shown as keys and misplaced keys in figure 6.1, a missing level of encryption in the message sent by S to A in section 7.5.3, and several missing sets of parentheses to control the scope of the $\models$ operator.

Appendix D. Acknowledgements

Thanks are due to my supervisor, Roger Needham, for his assistance with my work, and to Andy Hopper at ORL (then Olivetti Research) for the CASE sponsorship I received from ORL.

Particular thanks are due to Gray Girling, my mentor at ORL, for his excellent constructive criticism and, where necessary, vigorous application of motivation !

Thanks to Aldabra Stoddart for tea, sympathy and creamycakes. She has been very helpful, sympathetic and even co-conspiratorial, and lately an inspiration due to her recent submission of her own thesis.

Thanks again to Aldabra and also to Austin Donnelly and Karl Ewald for their willingness to read my accumulated prose and comment on it.

Thanks also to all my other friends, who've been variously encouraging, disbelieving and indifferent, but in whose company I was able to escape from reality; thanks also to the contributors to the Debian Project for firstly providing an invaluable displacement activity and lately tolerating my sudden absence.

Finally, thanks of course to my partner, Marisa Lohr, for constant encouragement, support, tolerance, and lately proofreading. Without her I might never have completed this thesis at all.

Bibliography

[AN96]
M. Abadi, R.M. Needham. *Prudent engineering practice for cryptographic protocols.* IEEE Trans. on Software Engineering **22**/1, 6–15, January 1996.

[BAN89]
M. Burrows, M. Abadi, R.M. Needham. *A Logic of Authentication.* Tech. Rep. 39 (December 1989), Digital Systems Research Center.

[BAN90]
M. Burrows, M. Abadi, R.M. Needham. *Rejoinder to Nessett*. Operating Systems Review **24**/2, 39–40, April 1990.

[Chaum81]
D. Chaum. *Untraceable electronic mail, return addresses, and digital pseudonyms.* Comm. ACM **24**/2, 84–88, 1981.

[Chaum88]
D. Chaum. *The Dining Cryptographers Problem: Unconditional Sender and Recipient Untraceability.* J. Cryptology **1**/1, 65–75, 1988.

[Churchyard97]
H. Churchyard. *Singular 'their' in Jane Austen and elsewhere.* `http://uts.cc.utexas.edu/~churchh/austheir.html`.

[Gutmann97]
P. Guttmann. *cryptlib.* `http://www.cs.auckland.ac.nz/~pgut001/cryptlib/`.

[Jackson96]
I.W. Jackson. *Anonymous addresses and confidentiality of location*, in 1st International workshop on Information Hiding (Cambridge UK) pages 115–120 May/June 1996.

[LSP82]
L. Lamport, R. Shostak, M. Pease. *The Byzantine generals problem.* ACM Trans. on Programming Languages and Systems **4**/3, 382–401, July 1982.

[Liebl93]
A. Liebl. *Authentication in distributed systems: a bibliography*. Operating Systems Review **27**/4, 31–41, October 1993.

[Loki95]
L. Cottrell. *Mixmaster and Remailer Attacks.* `http://www.obscura.com/~loki/remailer/remailer-essay.html`.

[Lowe95]
G. Lowe. *An attack on the Needham-Schroeder public-key authentication protocol.* Information Processing Letters **56**/3, 131–133, 1995.

[Meadows94]
C. Meadows. *A model of computation for the NRL protocol analyser*, in IEEE Computer Security Foundations Workshop VII, pages 84–89, June 1994.

[NS78]
R.M. Needham, M.D. Schroeder. *Using encryption for authentication in large networks of computers*. Comm. ACM **21**/12, 993–999, December 1978.

[Nessett90]
D.M. Nessett. *A Critique of the Burrows, Abadi and Needham Logic*. Operating Systems Review **24**/2, 35–38, April 1990.

[OED89]
Oxford English Dictionary. Oxford University Press. 2nd ed, 1989.

[ORL92]
R. Want, A. Hopper, V. Falcao, J. Gibbons. *The Active Badge location system.* ACM Trans. on Information Systems **10**/1, 91–102, January 1992.

[Ousterhout93]
J.K. Ousterhout. *An Introduction to Tcl and Tk*. Addison-Wesley, 1993.

[PP89]
B. Pfitzmann, A. Pfitzmann. *How To Break The Direct RSA Implementation of Mixes*, in Advances in Cryptology — Eurocrypt '89, pages 373–381.

[PW87]
A. Pfitzmann, M. Waidner. *Networks Without User Observability.* Computers and Security **6**/2, 158–166, February 1987.

[RB94]
M.K. Reiter, K.P. Birman. *How to securely replicate services.* ACM Trans. on Programming Languages and Systems **16**/3, 986–1009, May 1994.

[RSA78]
R. Rivest, A. Shamir, L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems.* Comm. ACM **21**/2, 120–126, February 1978.

[Rahnema93]
M. Rahnema. *Overview of the GSM System and Protocol Architecture.* IEEE Communications Magazine, April 1993.

[SMA95]
D. Samfat, R. Molva, N. Asokan. *Untraceability in Mobile Networks*, in MOBICOM '95 (Berkeley, CA), pages 111–118. ACM, November 1995.

[Shamir79]
A. Shamir. *How to share a secret*. Comm. ACM **22**/11, 612–613, November 1979.

[Simmons88]
G. Simmons. *How to (really) share a secret*, in Advances in Cryptology — Eurocrypt '88.

[Syverson90]
P.F. Syverson. *Formal semantic for logics of cryptographic protocols*, in IEEE Computer Security Foundations Workshop III, pages 32–41, June 1990.

[Waidner89]
M. Waidner. *Unconditional Sender and Recipient Untraceability in Spite of Active Attacks*, in Advances in Cryptology — Eurocrypt '89, pages 302–319.

[babel96]
C. Gülcü, G. Tsudik. *Mixing Email with Babel*, in Internet Society Security Workshop, 1996.

[cdpd93]
Cellular Digital Packet Data (CDPD) System Specification. Release 1.0, July 1993.

[kerberos87]
S.P. Miller, C. Neuman, C.I. Schiller, J.H. Saltzer. *Kerberos Authentication and Authorisation System*, in Project Athena Technical Plan, section E.2.1. MIT, July 1987.

[mobip96]
Analysis of Security and Privacy in Mobile IP, in 4th International conf. on Telecommunications Systems Modelling and Analysis (Nashville, USA), March 1996.

[onion96]
D.M. Goldschlag, M.G. Reed, P.F. Syverson. *Hiding Routing Information*, in 1st International workshop on Information Hiding (Cambridge, UK), pages 137–150, May/June 1996.

[onion97]
P.F. Syverson, D.M. Goldschlag, M.G. Reed. *Anonymous Connections and Onion Routing*, in IEEE Symposium on Security and Privacy, 1997.