LINK Personal Communications Programme

Third Generation Mobile Telecommunications Systems Security Studies

Technical Report 2:

Security Mechanisms for Third Generation Systems

Final version

15 May 1996

Communications Security and Advanced Development Group,
Vodafone Ltd.

Telecommunications Systems Group,
GPT Ltd.

Information Security Group,
Royal Holloway, University of London.

Document Release

Document:

Technical Report 2:
Security Mechanisms for Third Generation Systems
(Final version)

Responsible Partner:

RHUL.

Contributors:

Dr. Jason Brown (Vodafone)
Dr. Liqun Chen (RHUL)
Mr. Dirk Goj (RHUL)
Dr. Dieter Gollmann (RHUL)
Mr. Yong-Fei Han (RHUL)
Dr. Nigel Jefferies (Vodafone)
Prof. Christopher Mitchell (RHUL)
Prof. Michael Walker (Vodafone)
Dr. Dale Youngs (Vodafone)

Approved for Distribution:

..... Prof. Christopher Mitchell
(Partner Manager)

..... Prof. Michael Walker
(Project Liaison Officer)

0. Executive summary

This report is the second deliverable from the DTI/EPSRC LINK Personal Communications Programme project *'Security Studies for Third Generation Mobile Telecommunications Systems'*. The key objectives of the project may be summarised as follows:

- to identify the range and type of security features which third generation systems may be expected to support;
- to propose detailed guidelines on the classes of mechanisms that could be used to provide the identified security features;
- to define the infrastructure needed for the provision, operation and management of the security features;
- to assess the extent to which the security features, mechanisms and infrastructure elements need to be standardised.

During the first year of the project (February 1993-January 1994) the work of the project was dominated by major contributions to, and the continuing influence of, standardisation activity within ETSI SMG5 (UMTS) and ITU TG 8/1 (FPLMTS). This work culminated in the production of the first version of Technical Report 1, [1], the first project deliverable, dealing with the first of the above projective objectives.

During 1994 and 1995 the project moved on to a more intensive study of the second of the project objectives, namely a consideration of security mechanisms to support various of the security features required in third generation mobile telecommunications systems (in parallel with this work, second and third (and final) versions of Technical Report 1 were produced, [2,3]). This report is based on this work, and is the third and final version of a report considering in detail how security mechanisms can be used to provide some of the most problematic of the security features required for third generation systems.

The report begins, in Section 2, with a brief summary of the security features identified in Technical Report 1, [3], as being the most important for third generation systems. Section 3 gives a general introduction to security mechanisms, and is intended to provide a general motivation for the specific security mechanisms considered in detail in subsequent sections of this report. Section 4 contains the results of a general investigation of entity authentication techniques, and in particular a general examination of some of the design issues likely to be relevant in mobile telecommunications networks. This is immediately followed in Section 5 by the results of a study of possible mechanisms for the provision of service-related authentication in third generation systems. In Section 6, the problem of integrating encryption with the multiple access method employed on the air interface is considered; given the likely use of direct sequence CDMA for third generation systems, the possibility of combining encryption with CDMA modulation is considered in detail. Section 7 is concerned with possible key management techniques to support end-to-end security features. In Section 8 terminal-related security issues are considered, and in Section 9 the issue of privacy for user identity and location information is discussed in some detail. The last main part of the report is Section 10, which is concerned with the application of formal techniques to evaluate some of the mechanisms proposed elsewhere in this report. The report concludes in Section 11 with a review of some of the main areas for further research identified during the life of the project.

1. Introduction

The main purpose of this document is to consider the types of security mechanisms appropriate for providing certain security features to be supported by third-generation mobile telecommunications systems (3GS). The features considered are those whose provision appears likely to be most difficult. This includes those security features either for which the security mechanisms in use in first and second generation mobile telecommunications systems may not be appropriate, or which are not provided in existing systems.

The main security features for which security techniques are examined in detail are as follows:

- service-related authentication, i.e. authentication of entities directly involved in the provision of service to one or more users,
- encryption on the air-interface (and the relationship of encryption to the multiple access method),
- key management for end-to-end security features,
- identity and location privacy for the mobile user, and
- terminal-related security.

We also consider the issue of formal verification of authentication and key distribution mechanisms.

Before proceeding we now describe the structure of this document in more detail, describing the contents and status of each section in turn.

To put the work described in this report into context, in Section 2 we briefly review the security features likely to be required for 3GS. A much more detailed discussion of security features likely to be appropriate for 3GS can be found in 3GS3 Technical Report 1, [3].

In Section 3 we make some general observations about security mechanisms for 3GS, including providing some criteria for evaluating them. We also review what types of mechanism are available, and discuss priorities for different classes of mechanism.

The main part of this report commences in Section 4 with a general discussion of the provision of entity authentication. We start by reviewing the types of approach which could be followed, including a brief summary of the approaches followed by the ETSI GSM and DECT systems. We then move on to an analysis of how the environment for an entity authentication protocol affects its design, where by the environment we include: the processing and storage resources available to the entities, the political restrictions on the types of cryptographic technique which can be used, the nature of the communications channels in use, and the trust relationships between entities.

Section 5 follows naturally from Section 4 in that we look at a particular application of authentication techniques; specifically in Section 5 we focus on the issue of providing service-related authentication in 3GS. We look at both the role and functional models for 3GS (described in TR1, [3]), and see what options exist for providing authentication with respect to these models. We then give examples of protocols meeting the various options, and finally we show how these example protocols can be adapted to provide session key establishment across the air interface.

Encryption of user and signalling data sent across the air interface is the main concern of Section 6. In particular we consider the interaction between data encipherment and the choice of multiple access method. After a review of the main candidates for managing multiple access (TDMA and CDMA), and a brief review of encipherment of TDMA data, we consider the possibility of providing data confidentiality by manipulating CDMA sequences. This in turn has motivated a study of the selection of appropriate CDMA sequences, and a discussion is given of the relative merits of designed versus random sequences.

Section 7 is concerned with the problem of key management to support end-to-end security features (as opposed to key distribution for air interface features, as discussed in Section 4.5). We start by describing a novel solution to the problem of simultaneously providing keys for end-to-end data

encipherment between mobile users, and at the same time allowing legally-controllable access to data by government agencies (e.g. for law enforcement). The described solution is based on the use of trusted third parties. We also look at the case where the trusted third parties may not be individually trustworthy, and propose a solution addressing potential user needs for distributing trust. On a rather different note, we go on to consider the possible application of information-theoretic schemes for key agreement between remote entities, which do not require the intervention of any third parties or the pre-establishment of secret keys.

In section 8, concerned with terminal-related security issues, we describe the possible threats to the security of user terminals, and briefly review some possible approaches to meeting these security needs.

Section 9 deals with the issue of identity and location privacy for mobile users. The problem is discussed in some detail, and an attempt is made to classify solutions. The remainder of this section is concerned with example solutions, each of which are discussed and analysed.

The final main part of this report, Section 10, is concerned with formal analysis of security mechanisms. Because of the highly specialised nature of the research in this area, this section is extremely brief, and merely refers to a separate technical report which contains the detailed analyses.

The report concludes with an overview of the main areas for further research (Section 11), a bibliography (Section 12), and, last of all, a table of abbreviations in Section 13.

2. Security features

A detailed study of potential security features for 3GS can be found in 3GS3 Technical Report 1, [3], referred to throughout this report as TR1. For the purposes of this document we list the 19 security features described in subsection 7.4.4 of [3] as being of greatest relevance to 3GS. The intention of reiterating them here is to set the remainder of this report in its proper context.

Each of the 19 identified security features is assigned to one of the following categories:

- Confidentiality,
- Integrity,
- Authentication,
- Non-repudiation,
- Access Control,
- Security of management,
- Management of security, and
- Supplementary.

2.1 Confidentiality

1. Confidentiality of signalling and control data. This feature ensures that the signalling and control data are not made available or disclosed to unauthorised parties.

2. Confidentiality of user traffic. This feature protects against unauthorised eavesdropping on user traffic.

2.2 Integrity

3. Integrity of signalling and control data. This feature provides protection against manipulation (modification, insertion and/or replay) by unauthorised parties of signalling data or control data.

4. Integrity of user traffic. This feature protects against manipulation (modification, insertion and/or replay) by unauthorised parties of user data on the radio path or in the fixed network.

2.3 Authentication

5. Service-related authentication (or **Authentication among users, service providers and network operators**). This feature provides corroboration of the identities of a user, corresponding service provider and network operator underlying the provision of service to a user.

> Note that the name *service related authentication* is preferred because it covers authentication relating to the terminal.

6. Authentication between network operators/service providers (or **Management related authentication**)**.** This feature provides corroboration of the identity of one service provider or network operator to another.

> The name *authentication between network operators/service providers* seems more natural, but it needs to be distinguished from feature number 5.

7. Message origin authentication. This feature provides verification that transmitted signalling data, control data and/or user traffic originates from the claimed entity.

2.4 Non-repudiation

8. Non-repudiation of origin and delivery of transmitted data. This feature provides proof to a third party that a message was sent or received by a certain entity.

9. Non-repudiation of access to stored data. This feature provides protection against an entity denying having attempted to access the stored data.

10. Non-repudiation of access to a service. This feature provides protection against an entity denying having attempted to access a service.

2.5 Access control

11. Access control to a facility. This feature ensures that a UIM (User Identity Module) or terminal equipment can only be used by an authorised party.

12. Access control to a service. This feature ensures that only authorised parties can access a service. *Note that the question of how this feature will be supported by **Service-related authentication** is marked in TR1, [3], as being for further study.*

13. Access control to stored data. This feature ensures that only authorised parties can access stored data.

2.6 Security of management

14. User event reports. This feature ensures that a user will receive warning announcements or indications at critical moments in the operation of services.

15. Event logging and recording. This feature ensures that a service provider can log and record activities relating to a user or subscriber.

16. Event recovery. This feature ensures that a service provider can restore data relating to a user or subscriber upon failure, trace a particular user and refuse access to services requested by a particular user.

2.7 Management of security

17. Key management. This feature ensures that a service provider and/or network operator can manage keys used for the establishment of a private control and communication channel between the network operator and user.

18. Cryptographic information management. This feature ensures that a service provider and/or network operator can manage the generation, distribution, storage, updating and deletion of cryptographic information to assure the integrity of this information.

2.8 Supplementary

19. Support for end-to-end security service. This feature ensures that the service provider/network operator can provide end-to-end security services to particular fixed or mobile users, subject to the availability of additional end user equipment.

3. Security mechanisms

3.1 Introduction

In this section we make some general observations about the selection of security mechanisms to support security features for third generation systems. In subsection 3.2 we give some general criteria for evaluating the suitability of security mechanisms. This is followed, in subsection 3.3, by a description of the range of mechanisms available for use in 3GS. This section concludes, in subsection 3.4, with an analysis of the selection and prioritisation of security mechanisms for 3GS.

3.2 Criteria for evaluating security mechanisms

3.2.1 Introduction

In order to compare different security mechanisms which might be used within a 3GS architecture, criteria are needed by which mechanisms can be judged. Given an agreed set of 'objective' criteria, a judgement can be made as to which mechanisms are preferable. In this subsection some possible criteria are proposed and described.

3.2.2 Possible criteria

Potentially useful criteria are listed here in no particular order. However, some attempt has been made to classify these criteria.

3.2.2.1 Security service provision

1. **Fitness for purpose**. Most fundamentally, the security mechanisms employed must provide the security services which they are designed to provide. This should be possible to establish using informal arguments.

2. **Security proof**. If possible, the fitness of the security mechanisms should be established using formal (mathematical) techniques.

3. **Algorithm maturity and exposure**. Long-term existence of a (non-discredited) algorithm in the public domain may be an advantage since it will necessarily have resisted cryptanalytic attack.

4. **Availability of replacements**. The availability of similar replacements for a mechanism (in the event of it being discredited or otherwise rendered unusable) is a significant advantage.

3.2.2.2 Communications overheads

1. **Numbers of messages**. All else being equal, security mechanisms are to be preferred which minimise the number of messages which need to be exchanged.

2. **Total lengths of messages**. Security provision will inevitably involve transferring additional information across communication links. Clearly, minimising the total amount of such information transfer is desirable.

3. **Message expansion**. Apart from one-off overheads, some security mechanisms (e.g. encryption) involve expanding message content by a fixed ratio. Clearly, where bandwidth is at a premium, such as on the radio path, such overheads need to be minimised.

4. **Performance effects**. Certain types of security mechanism may degrade the quality of a channel. For example, use of encryption may actually magnify the effects of bit errors, i.e. the channel bit error rate may be increased. It would clearly be desirable to minimise any such impairment of communications channels.

3.2.2.3 Administration overheads

1. **Key storage**. Certain cryptographic algorithms require the storage of relatively large amounts of key material. Mechanisms which minimise key storage may have very significant advantages.

2. **Storage of other security parameters**. Use of certain types of security mechanisms will require the storage of other types of information. For example, some authentication mechanisms require communicating parties to store sequence numbers for every other party with which they communicate. Other authentication mechanisms require all 'recently received' messages to be stored to detect malicious replays occurring within the tolerance interval for synchronised clocks.

3. **Need for security servers**. Some mechanisms may require the participation of a specific security server (either on-line or off-line). For example, authentication mechanisms may require an on-line authentication server, an off-line certification authority or an on-line trusted time server (to provide clock synchronisation).

4. **Involvement of other entities**. A mechanism to provide a security feature between two entities may involve further entities. The number of such additional entities should be minimised, and the necessary level of trust in such entities should also be minimised.

3.2.2.4 Processing and other hardware overheads

1. **Cryptographic algorithm calculation**. Many security mechanisms will require entities within the system to perform cryptographic calculations. In some cases, the amount of processing required could have significant cost and/or time delay ramifications (e.g. implementing RSA in a user token). Hence minimising cryptographic complexity is a desirable goal.

2. **Other computation**. For similar reasons it would be desirable to minimise any other computations relating to the provision of security services.

3. **Special hardware needs**. Some mechanisms require the presence of particular functionality at communicating entities. For example, some authentication mechanisms require closely synchronised clocks. Other authentication mechanisms, and some key management mechanisms, require the means to generate either genuine random values or unpredictable pseudo-random numbers. Minimising such requirements would clearly be desirable.

4. **Matching processing requirements**. Different security mechanisms distribute processing requirements differently between sets of communicating entities. Ideally a security mechanism will match the processing requirements to the capabilities available to the various entities. Moreover, different mechanisms have varying proportions of pre-processing, 'real-time' processing, and post-processing. For example, one authentication mechanism may allow more pre-processing than another, speeding up an on-line transaction. This 'time distribution' of processing requirements must also be taken into account in mechanism choice.

3.2.2.5 Adherence to international standards

1. **ISO/IEC SC27 Mechanism Standards**. Where possible, it would be desirable for the mechanisms employed to adhere to ISO standards produced by ISO/IEC SC27, e.g. ISO/IEC 9798 (authentication mechanisms), [4], ISO/IEC 11770 (key management), [5].

2. **OSI Security Architecture and Security Frameworks**. Where relevant it would be desirable for the security mechanisms to be employed in a way conforming with the OSI Security Architecture (ISO 7498-2, [6]) and the multi-part security framework standard (ISO/IEC 10181, [7]).

3.2.2.6 Limitations on use

1. **Existence of patents**. The existence of patents on a mechanism, whether national, regional or world-wide and whether applied for or granted, can be a significant disadvantage.

2. **Export restrictions**. Some mechanisms may be subject to widespread export restrictions. The existence of such restrictions will be a clear disadvantage.

3.3 Security mechanisms available for 3GS

The text in this section is based on a paper, [8], prepared for publication as part of the 3GS3 project.

3.3.1 Introduction

We next consider the range of security techniques available to future designers and implementors of mobile telecommunications networks. Whilst our primary focus is mobile networks, the techniques described are of general applicability in telecommunications.

Security provisions in systems are present, not for their own sake, but to combat identified security *threats*. To combat these threats requires the provision of specific security *features* (sometimes known as *services*), such as

- *confidentiality* - to address the threat of unauthorised disclosure of information by means of eavesdropping, etc.,
- *data integrity* - to address the threat of unauthorised modification to information,
- *origin authentication* - to address the threat of information being spuriously inserted into a network,
- *entity authentication* - to address the threat of one entity masquerading as another,
- *non-repudiation* - to address the threat of an entity repudiating its actions (i.e. denying actions it has taken).

These features exist as abstract concepts, and are independent of the means used to provide them. Features are provided by security *mechanisms* (or *techniques*), which include

- *encipherment algorithms* - used to help provide confidentiality features,
- *integrity mechanisms* - (which include the well-known MACs), used to help provide data integrity and origin authentication features,
- *digital signature algorithms* -which can be used to help provide non-repudiation features,
- *authentication exchanges* - used to help provide entity authentication features.

At this point we should mention ISO 7498-2: 1989, [6], which provides a standardised language for discussing security features (or security services as ISO 7498-2 describes them) and mechanisms. We will use the terminology defined in ISO 7498-2 wherever possible, although we prefer to use the term security feature rather than security service to avoid confusion with services provided via a telecommunications network. ISO 7498-2 defines a range of types of security feature, as well as a classification of security mechanisms, and describes which mechanisms might be used to provide each of the defined security features.

In looking at techniques appropriate to telecommunications networks we will summarise the progress which has been made in recent years in providing general-purpose standards for security mechanisms. This work has primarily taken place within ISO/IEC JTC1/SC27/WG2.

3.3.2 Security mechanisms

We now consider a range of types of security technique which are relevant to the provision of the types of security feature discussed in Section 2 above. In particular we consider the following categories of security mechanism (note that some techniques can be used to help as components in other, more complex, mechanisms):

- *encipherment* mechanisms (for providing *confidentiality* features),
- *integrity* mechanisms (for providing *data integrity* features and as a building block in authentication exchanges),

- *digital signature* mechanisms (for providing *data integrity* and *non-repudiation* features),
- *hash-functions* (used in conjunction with digital signatures and also as a means of building integrity mechanisms), and
- *authentication exchanges* (for providing *entity authentication* features).

GSM and DECT make use of encipherment mechanisms (to provide air interface data confidentiality and user identity confidentiality), integrity mechanisms (to help construct authentication exchanges), and authentication exchanges (to provide entity authentication).

We now look at each of these categories of mechanism in a little more detail.

3.3.3 Encipherment techniques

No international standards exist for encipherment techniques, despite the US standardisation of the *DES* block cipher algorithm some 15 years ago (ANSI X3.92-1981), and its subsequent adoption as a de facto standard by the international banking community. Instead work has focused on creating an international *register* of encipherment algorithms, which will provide each registered algorithm with a unique name. The form of register entries, and the procedure for having entries added to the register, is standardised in ISO/IEC 9979: 1991, [12]. The international *Registration Authority* is the NCC here in the UK.

The GSM standards make use of a proprietary algorithm (called *A5*), which is owned by ETSI, and the details of which are kept secret, and released only to authorised manufacturers who are obliged to sign non-disclosure agreements. The A5 algorithm is not currently on the register (as of November 1994 there were ten registered algorithms), although there is no reason why it should not be, since registration of an algorithm does not require any disclosure of the algorithm's operation. Given the status of international standardisation for encipherment techniques, and the long-standing political sensitivity of encipherment technology, the GSM approach appears to be the only sensible way ahead, and for the foreseeable future this appears to be the most appropriate means for selection of encipherment techniques.

Before proceeding it is worth observing that, although there are no internal encipherment algorithm standards, there is a standard governing ways in which a block cipher algorithm (such as DES) might be used. This *modes of operation* standard (ISO/IEC 10116: 1991, [11]), has evolved from the ANSI standard prescribing modes of use for the DES block cipher (ANSI X3.106-1983). However, this ISO/IEC standard is probably of limited relevance to telecommunications networks, since high-speed stream ciphers rather than block ciphers are probably appropriate for the majority of telecommunications networks (it is certainly true that A5 is a stream cipher algorithm).

3.3.4 Integrity techniques

The purpose of a data integrity technique is to enable the recipient of a data string (protected using the technique) to verify both its origin and that it has not been changed in transit. Hence such a mechanism can be used to provide both data integrity and origin authentication features.

Standards for integrity mechanisms date back to the early 1980s; ANSI has published two integrity mechanisms for banking use (X9.9-1986 (revised) and X9.19-1985). A corresponding international banking standard, ISO 8731-1: 1987, has subsequently been published. All three of these standards specify use of the DES block cipher algorithm in Cipher Block Chaining (CBC) Mode to produce a *Message Authentication Code (MAC)*. A different integrity mechanism, called the *Message Authenticator Algorithm (MAA)*, is specified in the banking standard ISO 8731-2: 1992 (second edition). Following from the banking work, the international standard ISO/IEC 9797: 1994 (second edition), [13], for an integrity mechanism has been produced; this technique is also based on the use of a block cipher in CBC mode.

The GSM and DECT standards do not support the provision of origin authentication or data integrity services. However they both make use of an integrity mechanism as part of an authentication exchange technique which supports entity authentication of a mobile to a base station (we discuss this further below). Future telecommunications networks may well wish to offer a data integrity service as

an optional feature of service provision; it is also highly possible that ISO/IEC 9797 will not be appropriate for use, as it requires the implementation of a block cipher. However, one-way hash-function algorithms are at an advanced stage of standardisation (see below), and they can be simply adapted for use as integrity mechanisms, and this might well provide a possible route for future telecommunications networks. It certainly seems reasonable to suppose that future network designers will wish, wherever possible, to user standardised cryptographic techniques rather than design their own.

Finally it is important to note that, in parallel with the cryptographic *check functions* of the type described, the general class of integrity mechanisms also includes non-cryptographic mechanisms which are vital to the provision of integrity protection for sequences of data packets, for each of which an individual MAC or check-value may be computed. Examples of these non-cryptographic mechanisms include the use of sequence numbers or timestamps to provide data integrity for entire sequences of packets against manipulation (including threats such as duplication and deletion of entire packets).

3.3.5 Digital signature techniques

A digital signature technique is a function which, when applied to a data string, produces a result which enables the recipient to verify the origin and integrity of a data string, i.e. it can be used to provide data integrity and origin authentication features. Moreover it has the property that only the originator of a data string can produce a valid signature, i.e. being able to verify the correctness of a signature produced by an entity, does not provide the means to compute that entity's signature on any data string. Digital signature techniques can be used to provide *non-repudiation* of origin for a message, i.e. the recipient of a message with entity *A*'s signature on it, has evidence that *A* sent the message which even *A* cannot repudiate.

Digital signature algorithms can be divided into two types.

- Digital signatures *with message recovery* - which have the property that the message can be recovered from the signature itself; such signatures can normally only be applied to messages of limited length (e.g. of length at most 500 bits).
- Digital signatures *with appendix* - where the message needs to be sent in parallel with the signature, and signature acts as a `check' on the separately transmitted message.

A technique of the first type has been standardised in ISO/IEC 9796: 1991, [9]. The algorithm used is a variant of the well-known *RSA* algorithm, the first and best known of the *public key* (or *asymmetric*) cryptographic algorithms.

A US standard was recently published (the NIST *Digital Signature Standard (DSS)*) for a different signature algorithm (a variant of the El Gamal signature algorithm), this time of the second, and more generally useful, type. This technique is almost certain to be included in an emerging multi-part international standard, ISO/IEC 14888, [10], which will contain several techniques for Digital signatures with appendix. In parallel with these 'general purpose' standards, US and international banking standards committees have been developing standards incorporating the use of RSA signatures.

It is therefore now possible to say that there are a variety of well-established signature techniques available. However, they all share the same implementation difficulties (albeit to varying extents), namely that calculation of a digital signature can be very computationally intensive. This means that, until very recently, their implementation in hand-portable telecommunications terminals has been impractical. Smart cards capable of performing digital signatures in a small fraction of a second, which is what is needed, have proved difficult to make at an economic price.

However, as soon as such devices can be made at or close to the price of current smart cards, the use of digital signature techniques could become very attractive. As well as providing all the features that a conventional integrity mechanism, such as those built into GSM and DECT, can offer, key management for digital signature techniques is significantly simpler, since verification keys need not be kept secret. Moreover, digital signature techniques enable the provision of non-repudiation

features, which in turn makes telecommunications services such as *irrefutable charging* and/or non-repudiable financial transactions a possibility.

3.3.6 One-way hash-functions

One-way hash-functions are an essential component of all digital signature with appendix techniques. A hash-function takes as input a data string of arbitrary length, and outputs a *hash-code* of some small fixed length (e.g. 128 bits). In the context of a digital signature with appendix technique, the message to be signed is first input to a hash-function, and then the derived hash-code is input to the signature algorithm itself. However, hash-functions also have uses outside the context of digital signature algorithms. One important characteristic of hash-functions is that, unlike most other cryptographic functions, they do not use a secret key, and hence their entire operation is typically public.

A multi-part international standard for one-way hash-functions (ISO/IEC 10118, [14]) is under development. The first two parts (ISO/IEC 10118-1: 1994 and ISO/IEC 10118-2: 1994) have now been published. Part 1 contains general information, and Part 2 describes two methods for generating a hash-function from a block cipher.

Part 3, for which publication is planned in 1997, specifies in complete detail three 'dedicated' hash-functions, all designed specifically for the purpose. One, known as *SHS*, is already the subject of a recently published US NIST standard; it is designed to be used with the NIST DSS (signature) algorithm, although it is equally applicable to other signature algorithms. The other two, known as *RIPEMD-128* and *RIPEMD-160*, have been developed as part of the EU-funded RIPE project; the two RIPEMD functions are themselves 'improved' versions of MD4, a hash-function developed by RSA Inc. All three of these algorithms are simple to implement in software, and can process data at high rates even on modest microprocessors.

Part 4, also likely to be published in 1997, specifies a pair of hash-functions based on modular exponentiation. The reason for designing such hash-functions is that some digital signature algorithms (e.g. RSA) are also based on modular exponentiation, and hence a hash-function which could make use of the same arithmetic functions, perhaps implemented in hardware, might be very advantageous.

Parts 3 and 4 are most likely to be of value in telecommunications networks, in particular either as part of a digital signature algorithm, or as the basis of an integrity mechanism. In fact it is very simple to use a hash-function to produce an integrity mechanism; the data to be integrity protected is concatenated with a secret key, and the resulting data string is input to the hash-function. The hash-code output can then serve as a check-value on the original data string, and, given the hash-function is itself cryptographically strong, the check-value for a message can only be calculated by someone possessing the correct secret key.

Finally we note one other possible application for a one-way hash-function which may be of relevance to telecommunications networks. If a stored data object (e.g. a file) needs protection against change, it can be input to a hash-function and the output stored in a secure place. Recomputing the hash-code and comparing it with the stored value can be used to verify the correctness of the stored data. This is valuable in protecting against malicious changes made by malicious entities or programs (e.g. viruses).

3.3.7 Authentication exchanges

Authentication exchange techniques (or *authentication protocols* as they often called) are exchanges of cryptographically protected messages, which have the objective of enabling two communicating entities to verify one another's identity, i.e. they provide *entity authentication* features. Entity authentication can only be achieved for a single instant of time. Typically these exchanges are used to initiate a secure connection, and *Session Keys* may be established as a by-product of the authentication process.

In order for these authentication exchanges to work, some or all, of the messages need to be protected by a cryptographic mechanism, typically either an encipherment mechanism, a digital signature or an

integrity mechanism. In addition, non-cryptographic mechanisms (typically time-stamps or random 'nonces') need to be included in the messages to guarantee that the messages are 'fresh'.

The first standardised authentication exchanges were specified in CCITT X.509 (1988) (ISO 9594-8), which contains three different protocols all based on digital signature techniques; a revised version of this ITU-T Recommendation/ISO standard has recently been published. Since then a multi-part standard ISO/IEC 9798, [4], has been developed, containing a variety of different authentication exchange techniques using a range of different underlying cryptographic mechanisms.

Part 1 (ISO/IEC 9798-1: 1991) contains a general model for techniques of this type. Part 2 (ISO/IEC 9798-2: 1994) contains a set of authentication exchanges in which the messages are protected using an encipherment technique. Part 3 (ISO/IEC 9798-3: 1993) contains another set of authentication exchange techniques. These techniques are all based on the use of digital signature techniques to cryptographically protect the messages. Yet another set of authentication exchange techniques, this time based on the use of integrity mechanisms, is to be found in Part 4 (ISO/IEC 9798-4: 1995). In each of Parts 2, 3 and 4, some of the exchanges are suitable for use when synchronised clocks are available (and hence the messages contain time-stamps which can be used to verify their 'freshness'), and some use random or pseudo-random *nonces* (or *challenges*) instead.

The GSM and DECT standards both specify the use of authentication exchange mechanisms to provide entity authentication services between a mobile and a base station (over the air interface). In both cases, although the mechanisms are slightly different, the mechanisms conform to techniques specified in ISO/IEC 9798-4, i.e. the mechanisms are based on the use of integrity techniques. In both cases freshness is guaranteed by the use of pseudo-random nonces (challenges). The integrity techniques used are in both cases based on secret proprietary algorithms.

In both GSM and DECT the authentication exchanges are used at the start of a connection; they also provide the basis of session key distribution systems. The session key is subsequently used to encipher the data sent over the connection.

The use of authentication exchanges is fundamental to the provision of security in any situation where a communications link is liable to attack. It is difficult to imagine any future mobile networks not employing such a technique as a first line of defence against users wishing to fraudulently obtain service. Authentication exchanges may also be employed where different network operators an/or service providers need to verify one another's identity when they are collaborating to provide service to a user.

Finally it is also important to observe that authentication exchange techniques can be integrated with techniques to provide user identity confidentiality based on the use of continually updated 'temporary identities'. Such schemes already operate in GSM, and an elaborated version of the GSM and DECT schemes has recently been proposed for adoption as a mechanism for use in FPLMTS (see also subsection 9.3).

3.3.8 Summary

We conclude by observing that it should be clear that there already exist an increasingly wide range of standardised security techniques of potential use in future telecommunications networks. In the future the existence of these standardised techniques may remove the need for the design of new techniques whenever a new system is created. In the medium term, and perhaps sooner than many experts would like, this may well obviate the need for specialised cryptographers, at least as far as commercial communications are concerned! The real need, both now and in the future, is for skilled staff capable of selecting, integrating and managing the ever-increasing range of security products and systems available.

3.4 Selection and prioritisation of security mechanisms

We now describe a general approach for the selection of specific mechanisms for 3GS, and also for assigning priorities to the mechanisms. Various approaches exist for realising a particular type of mechanism, and the most common approaches are identified here.

The approaches identified are (generally) in accordance with the approaches identified by ISO and appropriate references are given in the text.

The priority assigned to a particular mechanism indicates the relative importance of the mechanism to 3GS (i.e. its impact on the secure operation of the system). A rating of primary (P) indicates that failure to use the mechanism could seriously impair secure operation of the system, whilst a rating of secondary (S) indicates that the consequences are (probably) less severe. A rating of tertiary (T) indicates that the specification of such mechanisms may be outside the scope of standardisation and need not be addressed. Further subdivision using numerals 1 - 3 accompanies the P or S rating (1 = most important, 3 = least important).

3.4.1 Authentication

Four approaches have been identified for providing authentication mechanisms (see ISO/IEC 9798, [4]):

- symmetric,
- public key,
- cryptographic check functions,
- zero knowledge.

Unless further study dictates otherwise, at least one mechanism should be realised for each of the above approaches. Mechanisms should, wherever possible, be based on those mechanisms used in current systems. Mechanisms should incorporate secure key distribution/agreement if necessary.

Priorities assigned to authentication are as follows:

- **P1** authentication of user to network operator/service provider;
- **P1** authentication of service provider/network operator;
- **S1** authentication between service providers and network operators;
- **S2** authentication of terminal to network operator/ terminal manager.

3.4.2 Confidentiality

The following approaches have been identified for providing confidentiality mechanisms:

- Block ciphers (and their various modes of operation). Block cipher modes of operation are specified in ISO/IEC 10116, [11]. Block ciphers can be divided into the following categories:
    - symmetric, and
    - asymmetric.
- Stream ciphers.

Note that ISO has defined a register for confidentiality mechanisms (or *encipherment* techniques), the form is which is specified in ISO/IEC 9979, [12].

Mechanisms for confidentiality may depend upon the choice of air-interface(s). Nevertheless, some aspects will be independent of this (e.g. management of cipher keys) and can be studied immediately.

Priorities assigned to confidentiality are as follows:

- **P2** confidentiality of user traffic, signalling data, and control data over the air interface;
- **P3** confidentiality of user traffic, signalling data, and control data sent between provider domains;
- **T** confidentiality of user traffic, signalling data, and control data sent within a provider domain;
- **T** confidentiality of stored data;
- **T** end-to-end user traffic confidentiality.

3.4.3 Anonymity

Three approaches have been identified for providing anonymity mechanisms:

- Identity confidentiality mechanisms, which can be divided into the following categories:
  - symmetric,
  - asymmetric,
- temporary identities,
- anonymous access.

Priorities assigned to anonymity are as follows:

- **P3** user anonymity.

3.4.4 Access control

Six approaches have been identified for providing access control mechanisms:

- non-cryptographic authentication techniques. These can be further subdivided into:
  - personal identification numbers,
  - simple challenge - response,
  - biometrics,
- registration,
- type approval,
- barring,
- auditing,
- physical means.

Priorities assigned to access control are as follows:

- **S1** access control to equipment;
- **T** access control to stored data

3.4.5 Integrity

Three approaches have been identified for providing integrity mechanisms:

- non-cryptographic integrity (error detection/correction, CRCs),
- cryptographic check functions (including MACs),

- hash functions.

A cryptographic check function is specified in ISO/IEC 9797, [13]. Cryptographic hash-functions are specified in ISO/IEC 10118, [14].

Priorities assigned to integrity are as follows:

- **S2** integrity of user traffic, signalling data, and control data over the air interface;
- **S2** integrity of user traffic, signalling data, and control data sent between provider domains;
- **T** integrity of user traffic, signalling data, and control data sent within a provider domain;
- **T** integrity of stored data;
- **T** end-to-end user traffic integrity.

3.4.6 Non-repudiation

Two approaches have been identified for providing non-repudiation mechanisms:

- symmetric;
- asymmetric.

Non-repudiation mechanisms are specified in ISO/IEC 13888, [15]. Priorities assigned to non-repudiation are as follows:

- **S3** non-repudiation of user access to services;
- **T** non-repudiation of access to stored data;
- **T** non-repudiation of origin and delivery of user traffic.

4. The design of authentication protocols

4.1 Introduction

In this section we consider some general issues relating to the design of authentication protocols. This serves as a preface to Section 5, in which we consider the specific problem of designing authentication protocols for use in providing service-related authentication services in future mobile telecommunications networks.

We start, in subsection 4.2, by giving a general analysis of the types of mechanism which can be used to provide entity authentication; this includes a review of the approaches followed by the second generation systems GSM and DECT. In subsection 4.3 we consider how the environment in which an authentication protocol operates, and the properties of the mechanisms underlying the protocol, affect the design of the protocol itself. In subsection 4.4 we consider how multiple authentication servers can be used to deal with the situation where an entity may not trust any single server absolutely; this may be of relevance to the situation where a mobile user is roaming in networks where the user has no knowledge of the trustworthiness of the network operators it may be using to send or receive calls. The protocols described will only work correctly if more than half the servers bahave according to the protocol specifications. We conclude in subsection 4.5 by describing another protocol to provide entity authentication using multiple servers, where, in some circumstances, an entity need only trust a minority of them to behave correctly.

4.2 Providing authentication in telecommunications networks

4.2.1 Introduction

Before analysing in more detail the nature of authentication protocols in the context of 3GS (in subsections 4.3 and 4.4 below) we review some of the types of cryptographic mechanism which have been considered for providing user authentication in 3GS. This review is intended as a starting point for a systematic approach to the selection of entity authentication mechanisms for 3GS.

The scope is very restricted in that it only considers mechanisms for one type of authentication, namely authentication of a user to a service provider, for example as part of call set-up. Moreover, it reaches no conclusions or recommendations, but rather outlines a number of options for further investigation, and draws attention to a number of factors which need to be considered as part of that investigation.

4.2.2 Mechanisms for user authentication

This subsection outlines a number of approaches to providing a mechanism for user authentication. The approaches are given in outline only. Variations on all of them are possible, but these are not discussed. Potential advantages and disadvantages are listed, but these are open to debate and no assertion is made or implied that the lists are complete.

4.2.2.1 The GSM approach

This approach is based on the use of a symmetric authentication algorithm *A*, which is implemented in the user's access device and in an authentication centre controlled by the user's (home) service provider. It requires a user-specific authentication key *K*, which is held in the access device and the authentication centre.

The mechanism works as follows. When the user attempts to access a service, for example to register or to make or receive a call, the network operator recognises the service provider of the user from the user's identity and signals to that service provider requesting authentication data. The service provider then instructs the authentication centre to produce the data and signals this back to the network operator. The data consists of one or more pairs of challenges and responses, CHALL, RES. The response RES is computed from the challenge CHALL using the algorithm *A* under control of the user specific key *K*. The network operator then sends CHALL to the user and awaits a response XRES. This response is computed in the user's access device, from the challenge received from the operator using the algorithm *A* under control of *K*, and sent to the operator. Upon receipt, the operator checks that RES and XRES are the same. If they are not, authentication has failed and access is denied.

Typically, the network operator will request more than one set of data so that it does not have to signal back to the service provider at every call attempt. Each CHALL, RES pair is used once.

There are a number of advantages with this type of mechanism which need to be considered as part of the selection procedure for the most appropriate mechanism for 3GS.

Advantages are:

- The user-specific key *K* is known only to the user's service provider and the user's access device; it does not have to be revealed to any other party.
- The authentication algorithm *A* may be specific to the service provider, or indeed to the service provider and a particular group of its users. It does not have to be implemented or known by any other party, and the service provider can change it for one or more of its users by merely issuing a new access device. Only the sizes of the parameters CHALL and RES need to be standardised.
- The mechanism can be readily adapted to provide a means to generate encryption keys for protecting user traffic and signalling on the radio path, a fresh key being generated for each call.

Disadvantages are:

- The network operator always has to signal back to the service provider in order to obtain the authentication data, none of this data can be re-used and all is specific to the particular user.

4.2.2.2 The DECT approach

This is a modification of the GSM approach which removes the need for the operator to request user authentication data from the service provider for each access.

The approach is based on the use of two symmetric algorithms *A* and *B*, and a user specific authentication key *K*. Algorithms *A* and *B* are implemented in the user's access device, algorithm *A* is implemented in an authentication centre controlled by the user's (home) service provider, whilst algorithm *B* is implemented by every network operator, or at least by every network operator who provides telecommunications services on behalf of the user's service provider. The key *K* is held in the user's access device and in the service provider's authentication centre.

The mechanism works as follows. When the user first attempts to register with a network operator, the operator recognises the service provider for the user from the user's identity and signals to that service provider requesting an authentication key. The service provider then instructs the authentication centre to produce the key data and signals this back to the network operator. The key data consists of a key seed *S* and a key value *KS*. The key *KS* is computed from the seed *S* using the algorithm *A* under control of the user specific key *K*. The network operator authenticates the user as follows. It generates a challenge, CHALL, and computes a response RES from CHALL using algorithm *B* under control of the key *KS*. The user is sent CHALL along with the seed *S*, and the operator awaits a response XRES. This response is computed in the user's access device, by first computing *KS* from *S* using algorithm *A* under control of *K*, and then computing XRES from CHALL using algorithm *B* under control of *KS*. Upon receipt, the operator checks that RES and XRES are the same. If they are not, authentication has failed and access is denied. Note that the value of CHALL must change upon every use of this mechanism.

Advantages are:

- The pair *(S, KS)* may be used by the network operator for the particular user for as long as the user is registered with that network. Since the service provider chooses *S*, different operators or different registrations may be given different keys for authenticating the user.
- The user specific key *K* is known only to the user's service provider and the user's access device, it does not have to be revealed to any other party.
- The key generation algorithm *A* may be specific to the service provider, or indeed to the service provider and a particular group of its users. It does not have to be implemented or known by any other party, and the service provider can change it for one or more of its users by merely issuing a new access device.
- The mechanism can be readily adapted to provide a means to generate encryption keys for protecting user traffic and signalling on the radio path, a fresh key being generated for each call.

Disadvantages are:

- The network operator has to signal to the service provider in order to obtain a key *KS* which is specific to the particular user.
- The algorithm *B* must be known to and used by all operators that provide telecommunications services for the particular service provider. In practice the algorithm must be standardised.
- The service provider has to trust that the network operator will preserve the confidentiality of *KS*.
- A secure channel needs to be provided for the transfer of *KS* from the Service Provider to the Network Operator.

4.2.2.3 Digital signature approaches

This approach is based on the use of an asymmetric (public key) algorithm *A*, which is implemented in the user's access device and by every network operator, or at least by every network operator who provides telecommunications services on behalf of the user's service provider. Typically the algorithm would be RSA or the DSS. The user's access device contains the secret half K_s of a key for the algorithm, and the user's service provider holds the public half K_p of the key. The key K_s is used for signing, whilst the key K_p is used for checking signatures computed using K_s.

The mechanism works as follows. When the user first attempts to register with a network operator, the operator recognises the service provider for the user from the user's identity and signals to that service provider requesting the public key K_p for that user. There are two possible ways in which authentication may then proceed, as a bi-directional or as a unidirectional protocol.

1. With the bi-directional approach, the network operator sends a challenge CHALL to the user. This is received by the user's access device which uses *A* to compute a signature SIG on CHALL under control of the secret key K_s. The signature SIG is then sent to the network operator which uses algorithm *A* under control of the public key K_p to verify that it is the correct signature for CHALL for the particular user. The details of the checking depend upon the algorithm.

2. With the unidirectional approach, a signature SIG is computed by the user's access device using the algorithm *A* under control of K_s on some (user access device generated) 'timely' data TIM. The data TIM, SIG is sent to the operator. The operator checks the timeliness, or freshness, of TIM and then uses *A* under control of K_p to verify the signature SIG.

Advantages are:

- The key K_p may be used by the network for as long as the user is registered with that network.
- Network operators do not have to maintain confidentiality of the key K_p, and service providers do not have to entrust network operators with keys which could be used to make fraudulent accesses - knowledge of K_p does not yield knowledge of the signature key K_s.

Disadvantages are:

- Asymmetric algorithm signatures are long (512 - 1000 bits) in comparison with symmetric algorithm responses (typically 32 - 64 bits).
- Asymmetric algorithms are more complex than symmetric ones.
- Asymmetric algorithms are rare - probably only RSA and DSS are candidates, and RSA may not be politically acceptable - in practice the algorithm must be standardised.
- The network operator has to signal to the service provider to obtain the user specific public key K_p, and it has to have confidence in the authenticity of the key.
- The mechanism may not be readily adaptable to provide a means to generate encryption keys for protecting user traffic and signalling on the radio path, especially if the DSS algorithm is used.

4.2.2.4 Digital signatures with certified public keys

With this approach, the user's access device contains the asymmetric algorithm A and the user specific secret key K_s as with the mechanism described in the previous section. In addition, the device contains a certificate CERT for the user's public key K_p. Typically, CERT contains K_p, the user's identity, the identity of the service provider, expiry date, etc., and a digital signature SIG_sp. This signature, SIG_sp, is computed by the service provider on the other contents of the certificate using the algorithm *A* under control of the secret part SP-K_s of a service provider specific key. Algorithm *A* and the public half SP-K_p of the service provider's key are available to all operators which provide telecommunications services on behalf of the service provider.

The mechanism works as follows. When the user registers with the network operator, the access device sends the certificate CERT to that operator. The operator identifies the service provider from CERT and uses the service provider's public key SP-K_p to check the signature SIG_sp of the certificate. If this is correct, the operator accepts the public key of the user, and may then authenticate the user using one of the protocols outlined in the previous section.

Advantages (in addition to those listed in the previous section) are:

- The operator does not have to signal to the service provider for a user specific key - all it needs is the service provider's public key.

- The certificate may also be used to convey other constant, non-confidential but verifiable user related data which may be needed by network operators, thereby reducing the amount of signalling to the service provider.

Disadvantages are:

- Extra signalling over the radio path - CERT will typically be in excess of 1000 bits.

4.2.2.5 Identity based schemes

With this approach, the user has two identities, a public identity PI and a corresponding secret identity SI. These identities are constructed by the service provider and written to the user's access device. To construct the identities requires knowledge of some secret parameters (e.g. the factors of some large integer *N*), but the relationship between the identities can be verified using a parameter which does not need to be protected against disclosure (e.g. the integer *N* itself). The same parameters can be used by the service provider to compute the identities for all of its users. Verification of an identity uses a zero-knowledge protocol, which enables the verifier to be convinced that the access device knows the secret identity without the verifier (or any eavesdropper) gaining any knowledge of this identity - even if the verifier abuses the protocol.

The mechanism works as follows. When the user attempts to access a service, for example to register or to make or receive a call, the user's access device sends its public identity PI to the network operator. The network operator recognises the service provider of the user from the identity PI, and then executes the appropriate zero-knowledge protocol with the access device to verify that it knows the secret identity SI corresponding to PI.

Advantages are:

- The operator does not have to signal to the service provider for a user specific key - all it needs is the public parameter of the service provider. This parameter does not have to be kept confidential, but its authenticity does need to be assured.
- The public identity may also be used to convey other constant, non-confidential but verifiable user related data which may be needed by network operators, thereby reducing the amount of signalling to the service provider.
- The protocols are such that various parameters (e.g., the number of bits exchanged) can be adjusted to suit the level of security required (e.g. the probability of allowing access to a fraudulent user). Single (as well as multiple) iteration schemes are available. This feature of being able to quantify resources in terms of levels of security is attractive.
- Although more complex than symmetric schemes, zero-knowledge schemes appear to be less complex than public key schemes in terms of the mathematical operations that have to be performed.

Disadvantages are:

- Zero-knowledge protocols require the transfer of quite large amounts of data on the radio path (more than 1000 bits for the probability of accepting a fraudulent user to be less than 10^{-20}), although, as mentioned above, this can be adjusted to suit the level of confidence required.
- Suitable zero-knowledge protocols are, in comparison with symmetric algorithms, comparatively rare - although there is greater flexibility than with public key schemes. In practice the protocol would need to be standardised.
- The protocols cannot be used to carry confidential information - hence they cannot be used to establish encryption keys.

4.3 Tailoring authentication protocols to match underlying mechanisms

The text in this section is based on a paper, [16], prepared for publication as part of the 3GS3 project.

Authentication protocols are constructed using certain fundamental security mechanisms. We discuss here how the properties of the underlying mechanisms affect the design of authentication protocols. We firstly illustrate factors affecting the selection of protocols generally. These factors include the properties of the environment for authentication protocols, the resources of the authenticating entities, and the properties of the underlying mechanisms. We then consider a number of authentication protocols which are based on underlying mechanisms satisfying less stringent conditions than those required for the ISO/IEC 9798 protocols.

4.3.1 Introduction

This subsection of TR2 is concerned with the fundamental security mechanisms which are used to construct entity authentication protocols, and how the strength of these mechanisms affects the design of authentication protocols. For our purposes, the underlying mechanisms for an authentication protocol are divided into two categories, cryptographic mechanisms and time variant parameters (TVPs). The first category includes symmetric encryption, digital signatures, cryptographic check functions (CCFs) and zero knowledge techniques. The second category includes clocks for timestamping, nonce generators and sequence numbers. An authentication protocol, which provides mutual proof of identity and timeliness to a pair of communicating entities, is typically constructed using at least one underlying mechanism from both categories.

When a new authentication protocol is needed in a particular environment, the designer must first discover what underlying mechanisms are available. For implementation reasons there may be limits on available mechanisms, because each proposed underlying mechanism has particular properties corresponding to the particular environment in which the protocol works. In such a case, the protocol will need to be chosen to meet the needs of the environment and the 'strength' of the available mechanisms.

A variety of entity authentication protocols[1] have recently been standardised by the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). The four published parts of International Standard ISO/IEC 9798, [4], cover the general model (9798-1), authentication protocols based on symmetric encipherment algorithms (9798-2), authentication protocols based on public key algorithms (9798-3), and authentication protocols based on CCFs (9798-4). A fifth part of ISO/IEC 9798, covering authentication protocols using zero knowledge techniques, is currently at Working Draft (WD) stage. The ISO/IEC 9798 protocols use TVPs, e.g. timestamps, nonces and sequence numbers, to prevent valid authentication information from being accepted at a later time.

The protocols proposed in ISO/IEC 9798 have all been designed to use underlying mechanisms meeting rather stringent requirements, which may be difficult to meet in some practical environments. Hence here we look at alternatives to the ISO/IEC 9798 protocols which are still sound even when the underlying mechanisms do not meet such 'strong' conditions.

After a brief description of the relevant definitions and notation used in this subsection, the discussion starts in subsection 4.3.3 with an illustration of factors which affect protocol selection. These factors include the properties of the environment for an authentication protocol, the resources of the authenticating entities, and the properties of the underlying mechanisms. We then look at the ISO/IEC 9798 protocols in the context of requirements on the underlying mechanisms. In subsections 4.3.4, 4.3.5, 4.3.6, 4.3.7, 4.3.8 and 4.3.9, we consider possible alternatives to the ISO/IEC 9798 schemes, which do not make such strict requirements on the underlying mechanisms. Finally, in subsection 4.3.10, we give our conclusions.

[1] They are referred to as entity authentication mechanisms in ISO/IEC documents.

4.3.2 Definitions and notation

For the purposes of this subsection the definitions of general security-related terms described in the general model of ISO/IEC 9798, [4], apply. In addition the following definitions are used.

- *Multiple authentication servers:* entities who collaborate to provide an authentication service to authentication entities as clients.
- *Suppress-replay attack:* an attack which exploits a message by first suppressing it and replaying it later.

This subsection makes use of the following notation.

- A and B are the identifiers of entities A and B respectively (the distinction between an entity and its identifier should be clear from the context).
- K_{XY} is a secret key shared between entities X and Y.
- S_X is entity X's private signature transformation (we assume that the signature transformation does not provide message recovery).
- V_X is entity X's public verification transformation.
- E_X is entity X's public encipherment transformation (when used as part of an authentication protocol we assume that the encipherment operation will provide origin authentication and integrity services as well as confidentiality).
- D_X is entity X's private decipherment transformation.
- R_X is a nonce issued by X.
- T_X is a timestamp issued by X.
- N_X is a sequence number issued by X.
- $X||Y$ is the result of the concatenation of data items X and Y.
- Cert$X$ is a trusted third party's certificate issuing the public part of entity X's valid asymmetric key pair.
- $E_K(Z)$ is the result of the encipherment of data Z with a symmetric encipherment algorithm using the key K.
- $F_K(Z)$ is a cryptographic check value which is the result of applying the cryptographic check function F using as input a secret key K and an arbitrary data string Z.
- D_i is the ith optional text field.
- h is a one-way hash function.
- $A \rightarrow B$: m denotes that A sends message m to B.

4.3.3 Factors affecting protocol selection

In this section three important factors which affect the design of authentication protocols are discussed, namely the properties of the environment for protocols, the resources of the authenticating entities and the properties of the underlying mechanisms. As mentioned in Section 4.3.1, authentication protocols are constructed using underlying mechanisms which have varying security properties (informally, some are 'stronger' than others). The properties of the mechanisms, the environment in which the protocol is used and the resources which the authenticating entities have, all affect the design of an appropriate authentication protocol.

The discussion in this section illustrates that if the environment and entity resources for authentication satisfy less stringent conditions than those required for the ISO/IEC 9798 protocols, then the ISO/IEC 9798 protocols cannot be used, and a different protocol, tailored to match the properties of the

underlying mechanisms, needs to be designed. In subsequent sections we give examples of such variant protocols.

4.3.3.1 Properties of the environment for protocols

Before considering what underlying mechanisms are available and selecting a protocol which uses them, the designer must know the environment in which the protocol will work. A particular environment may impose very stringent requirements on the underlying mechanisms and the protocol itself. For the purposes of this subsection we consider the following aspects of the environment for an authentication protocol.

4.3.3.1.1 Communications channel

Communications channels are used to transmit the message exchanges during the authentication processing. The major property of a communications channel of interest here is what right each entity has for accessing the channel. The following three kinds of channels are possible instances:

- *Equal rights channel*, where all entities involved in the protocol, including intruders, have the same ability to access the channel. Everyone can send and receive messages. An intruder can intercept messages, replay messages with or without modification, and even introduce new fraudulent messages to the channel, but ***cannot*** suppress messages.
- *Authenticated channel*, where different entities have different rights to access the channel. One example of an authenticated channel is where the intruder can just intercept messages, but cannot modify messages or introduce fraudulent messages to the channel without being detected.
- *Non-authenticated channel*, where the intruder can not only intercept messages but also suppress messages, as well as insert, modify and replay messages.

Another property of a communications channel of interest here is whether it is a broadcast channel or a point-to-point channel.

- *Broadcast channel*, where there exist messages from a variety of different senders and/or for a variety of different receivers. Typically, to make a broadcast channel operate correctly, the sender's and/or receiver's names have to be sent across the channel.
- *Point-to-point channel*, where the channel is reserved for a particular sender and receiver who both know the initiator and terminator of each message, so that their names do not need to be sent across the channel.

4.3.3.1.2 Entity identifier

The major property of entity identities of interest here is which entities involved in the protocol are authorised to know a particular entity identifier during the authentication processing. We consider the following two possible situations.

- The identifier of an entity is allowed to be transferred in clear text. The majority of the protocols defined in ISO/IEC 9798 parts 2, 3 and 4, [4], are examples of protocols making this assumption, in that most of them require entities' *distinguishing identifiers* to be sent as part of the message exchanges. In fact, in the context of the ISO/IEC 9798 protocols a 'distinguishing identifier' may be a full identifying string for an individual person, or may only be a single bit used to distinguish one of two possible entities.
- The identifier of an entity is only allowed to be known to particular entities. One example of such an environment is a mobile telecommunications system in which a mobile user's identifier reveals the user's location and hence may need to be concealed from interceptors.

4.3.3.1.3 Trust relationship

The trust relationship describes whether one entity believes that the other entities involved in an authentication protocol will follow the protocol specifications correctly. The trust relationship of particular interest here is between authentication servers and clients. For instance, a client may not

trust an individual authentication server. Those protocols specified in ISO/IEC 9798 which make use of authentication servers assume that the server is trusted by both authenticating entities.

4.3.3.2 Resources of authenticating entities

For the purposes of this subsection we consider the following aspects of the resources of the authenticating entities.

4.3.3.2.1 Knowledge

An entity involved in an authentication protocol might have one or more of the following four kinds of knowledge before the authentication processing starts:

- shared *secret information* with another entity,
- *private information*, e.g. the private part of the entity's own asymmetric key pair,
- *reliable public information*, e.g. the public part of another entity's asymmetric key pair, or
- *no knowledge* of another entity.

4.3.3.2.2 Computational ability

The entities involved in an authentication protocol may or may not have the computational ability to perform the following operations:

- computation of complex cryptographic algorithms (e.g. a digital signature algorithm),
- generation of a key or an offset for a key randomly and securely, or
- generation of predictable or unpredictable nonces.

4.3.3.2.3 Time synchronisation

The entities involved in an authentication protocol may or may not have access to securely synchronised clocks or synchronised sequence numbers.

4.3.3.3 Properties of underlying mechanisms

In this subsection, we consider six properties of the mechanisms which are used to construct authentication protocols.

- *Trusted third parties.* There are a variety of authentication protocols which involve use of a trusted third party. For example, in a symmetric encryption or CCF based protocol, the claimant and verifier may share no secret before use of the authentication protocol, and in an asymmetric cryptography based protocol, the claimant and verifier may hold no public information for each other before authentication. In both cases a trusted third party can be used to set up appropriate keys. This third party is typically called an authentication server, and is trusted by both the entities involved.
- *Disclosure of the identifier of an entity.* In an authentication protocol based on symmetric encryption or CCFs, if one entity's distinguishing identifier has to be known to an entity involved before the authentication processing, it is necessary to send this distinguishing identifier unencrypted.
- *Abuse of digital signatures.* In an authentication protocol based on digital signatures with an unpredictable nonce as a challenge, it is possible for one entity to force the other to sign a particular text, which might then be used for malicious purposes perhaps in different contexts. This type of behaviour is sometimes known as using the signer as an 'oracle', and it is this type of attack which zero knowledge protocols are designed to prevent.

- *Disclosing corresponding plaintext/ciphertext pairs*. All the authentication protocols specified in Parts 2, 3 and 4 of ISO/IEC 9798 could give corresponding plaintext/ciphertext pairs to an interceptor.
- *Unpredictable and predictable nonces*. Nonces used in authentication protocols may be either predictable or unpredictable, that is a third party monitoring the messages corresponding to authentication exchanges may or may not be able to predict the nonces in advance. Those protocols specified in Parts 2, 3 and 4 of ISO/IEC 9798 which use nonces, do require that the nonces used are unpredictable to all third parties.
- *Using synchronised clocks*. Certain of the authentication protocols specified in Parts 2, 3 and 4 of ISO/IEC 9798 require the communicating parties to have synchronised clocks and depend on them for correctness.

We now discuss each of these properties of the underlying mechanisms in more detail. We also consider alternatives to the ISO/IEC 9798 protocols, which do not make such strict requirements on the underlying mechanisms.

4.3.4 Absence of trust in a third party

In this section we consider a situation where two entities, who share no secret based on symmetric cryptography or hold no public information based on asymmetric cryptography, want to complete unilateral or mutual authentication. Typically they will have to get assistance from a third party, referred to as an authentication server, during the authentication process. However, in some environments, these clients have no reason to trust an individual server, e.g. there may be malicious authentication servers who do not follow the protocol specifications correctly, [17].

In such a case the ISO/IEC 9798 protocols cannot be used directly. In order to design an authentication protocol which does not require trusting individual authentication servers, a range of possible approaches can be followed.

In the first approach, a client can choose which server is trustworthy and which is untrustworthy from a set of authentication servers, typically by applying a security policy or the history of performance and reliability. Yahalom et al., [18], proposed a protocol which allows a client or his agent to choose trustworthy authentication servers and avoid untrustworthy ones. One difficulty with this scheme is that a client may sometimes find it difficult to distinguish between trustworthy and untrustworthy servers.

The second approach uses many servers simultaneously to achieve authentication. Gong [17] proposed a protocol with multiple authentication servers such that a minority of malicious and colluding servers cannot compromise security or disrupt service. In that protocol two clients participate in choosing a session key, and each relevant server is responsible for converting and distributing a part of the session key. Two variations on this approach are described in Section 4.4 below, in both of which the servers each generate a part of a session key, which can be successfully established between a pair of entities as long as more than half the servers are trustworthy. Both schemes from Section 4.4 have the advantage of requiring considerably fewer messages than Gong's protocol.

The third approach, based on asymmetric cryptography, is to separate authentication information transfer from authentication information issue, i.e. to let the authentication information issuer be off-line. One instance of this approach is where a master server (sometimes called the authentication information issuer or certification authority) issues a certificate which is then held by another server (sometimes called the authentication information transferrer). The certificate is valid for a period of time, during which there is no need to further contact the master server, since the transferrer can provide the certificate. The client does not need to trust the transferrer who is on-line, but does need to trust the issuer who is off-line.

4.3.5 Entity identity privacy

First note that the issue of identity anonymity in security protocols is addressed in considerably more detail in Section 9 of this report. The remarks here serve as a general introduction to that section.

When an authentication protocol is used, the identities of the entities concerned may need to be sent across the communications channel, either embedded within or alongside the protocol messages. There are two main reasons why this might be necessary.

- Depending on the nature of the communications channel, all messages may need to have one or more addresses attached. More specifically, if a broadcast channel is being used, then, in order for a recipient of a message to know whether or not it is intended for them, a recipient address must be attached. In addition, many authentication protocols require the recipient of a protocol message to know the identity of the entity which (claims to have) originated it, so that it can be processed correctly (e.g. deciphered using the appropriate key). If this information is not available, as would typically be the case in a broadcast environment, then the originator address also needs to be attached.

- Certain authentication protocols, including some of those in ISO/IEC 9798, require the recipient's name to be included within the protected part of some of the messages, in order to protect the protocol against certain types of attack.

However, communicating entities may require a level of anonymity, which would prevent their name and/or address being sent across the channel (except in enciphered form). For example, in a mobile telecommunications environment it may be important for users that their identifiers are not sent in clear across the radio path, since that would reveal their location to an interceptor of the radio communications.

Of the two reasons listed for sending names across the channel, the second is usually simpler to deal with, since alternative protocols can typically be devised which do not require the inclusion of names in protocol messages; for example, as described in ISO/IEC 9798-4, 'unidirectional keys' can be employed. We therefore focus our attention here on the anonymity problems arising from the use of authentication protocols in broadcast environments. It is important to note that these anonymity problems are not dealt with in any published part of ISO/IEC 9798.

There are two main approaches to providing entity anonymity in broadcast channels:

- the use of temporary identities, which change at regular intervals, and

- the use of asymmetric (public key) encipherment techniques.

We now discuss three examples which make use of these approaches. In each case we consider a 'many-to-one' broadcast scenario, where many mobile users communicate with a single 'base' entity. In this case anonymity is typically only required for the mobile users, and, in any case, the mobile users can only receive from the base and hence there may be no need for the base address to be sent across the channel.

4.3.5.1 Example 1: GSM

First observe that in GSM, [40], temporary entity identities (called temporary mobile subscriber identities or TMSIs in GSM) are transmitted over the air interface instead of real entity identities (called international mobile subscriber identities or IMSIs in GSM). TMSIs are local identities and valid only in a given location area, and they are changed on each location update and on certain other occasions as defined by the network.

A mobile subscriber identifies himself by sending the old TMSI during each location update process, which has to be sent before authentication takes place and must therefore be sent unencrypted. However, the new TMSI is returned after authentication is completed and a new session key has been generated so that it can be, and is, transmitted encrypted. In the following three exceptional cases, the subscriber has to identify itself using its IMSI,

- initial location registration,

- old visitor location register is not reachable, and

- no old TMSI is available.

In these cases an intruder may be able to obtain the IMSI from the GSM air interface. After that, a new TMSI is allocated and returned encrypted.

4.3.5.2 Example 2: A protocol based on symmetric cryptography

We secondly consider a mutual authentication protocol, outlined in [19], which has been designed for possible adoption by UMTS and FPLMTS. Like the GSM solution, this scheme also uses temporary entity identities to provide identity and location privacy. In this protocol, temporary user identities are used at every authentication exchange including the case of a new registration, so that real user identities are never transmitted in clear text. This protocol is discussed in much greater detail in Section 9.3 of this report.

The entities involved in this protocol are one of number of users, *A*, a single 'base' entity *B* and an authentication server *S*.

The protocol provides the following security services to these principals.

- Mutual entity authentication between *A* and *B*.
- Identity confidentiality for *A* during the communications between *A* and *B*.
- Session key establishment between *A* and *B*.

The protocol makes use of five cryptographic check functions *Fi* ($1 \le i \le 5$). The expression $Fi_K(D)$ denotes the output of function *Fi* given key input *K* and data input *D*. The protocol also makes use of two types of temporary identities: S-identities shared by users *A* and the authentication server *S*, and B-identities shared by users *A* and the base entity *B*. These temporary identities are updated every time the authentication protocol is used.

In the protocol:

- I_A denotes the current S-identity of *A*;
- I'_A denotes the new S-identity of *A*;
- J_A denotes the current B-identity of *A*; and
- J'_A denotes the new B-identity of *A*.

There are two versions of the protocol, depending on whether or not *A* and *B* already share a valid temporary B-identity J_A and secret key K_{AB}.

4.3.5.2.1 Version 1: *A* and *B* share K_{AB} and J_A

The protocol has the following three steps:

M1: $A \rightarrow B: \; J_A || R_A$

M2: $B \rightarrow A: \; R_B || F4_{K_{AB}}(R_A) \oplus J'_A || F3_{K_{AB}}(R_B || R_A || J'_A)$

M3: $A \rightarrow B: \; F3_{K_{AB}}(R_A || R_B)$

where $\oplus$ denotes bit-wise exclusive-or of strings of bits, and session key

$$K'_{AB} = F5_{K_{AB}}(R_A || R_B || J'_A).$$

4.3.5.2.2 Version 2: *A* and *B* share no secret

For this version of the protocol we assume only that *A* and *S* share a secret key K_{AS} and a temporary identity I_A, and *B* and *S* have a secure channel which is available for exchanging messages **M2** and **M3**.

M1: $A \rightarrow B: \; I_A || R_A$

M2: $B \rightarrow S: \; I_A || R_A$

M3: $S \rightarrow B: \; F1_{K_{AS}}(R_A) \oplus I'_A || O || K_{AB} || F3_{K_{AS}}(I'_A || R_A || O)$

M4: $B \rightarrow A: \; F1_{K_{AS}}(R_A) \oplus I'_A || O || F3_{K_{AS}}(I'_A || R_A || O) ||$

$$R_B || F4_{K_{AB}}(R_A) \oplus J'_A || F3_{K_{AB}}(R_B || R_A || J'_A)$$

M5: $A \rightarrow B: \; F3_{K_{AB}}(R_A || R_B)$

where O is a key offset so that

$$K_{AB} = F2_{K_{AS}}(O || B) .$$

4.3.5.3 Example 3: A protocol based on asymmetric cryptography

Our third example is based on the use of asymmetric encipherment. In a protocol based on such techniques, the sender's identifier can be enciphered using the public part of the receiver's asymmetric key pair, so that only the receiver can recognise it. If the confidentiality of the receiver's identifier is required, a temporary receiver identifier can be used. We now describe an example of a protocol which works along these lines. This protocol is discussed in much greater detail in Section 9.4 of this report.

We first state the initial requirements.

1. *A* is one of number of users of a single broadcast channel, used for communicating with a single 'base' entity *B*.
2. *A* and *B* have their own personal asymmetric encipherment systems (E_A, D_A and E_B, D_B).
3. *A* and *B* have access to authenticated copies of the public encipherment transformation of one another.
4. *A*'s identifier has to be kept confidential.

The protocol then has the following three steps:

M1: $A \rightarrow B: \; E_B(A || R_A || I_A)$

M2: $B \rightarrow A: \; I_A || E_A(R_B || R_A || B)$

M3: $A \rightarrow B: \; E_B(R_A || R_B || A)$

where I_A is a temporary identifier for *A*. For this protocol to operate successfully, the nonces R_A and R_B must be incapable of being guessed by any third parties.

4.3.6 Avoiding abuse of digital signatures

During authentication processing based on digital signatures with 'unpredictable' nonces, entity *A* typically challenges entity *B* by sending a nonce R_A. *B* then sends *A* a signature-based message containing this nonce in reply to the challenge. *B* attaches no meaning to signing this 'random number'. However, by choosing the nonce appropriately *A* may be able to use *B*'s signature for malicious purposes.

The authentication protocols specified in ISO/IEC 9798-3 do discuss means of dealing with this possible problem, and to help avoid the worst consequences, a nonce issued by the signer is included

in the relevant signature. However, the same problem may still exist. We now consider the following three-pass authentication protocol given in Clause 5.2.2 of ISO/IEC 9798-3 (see also [20]).

M1: $B \rightarrow A$: $R_B || D_1$

M2: $A \rightarrow B$: $\text{Cert}A || R_A || R_B || B || D_3 || S_A(R_A || R_B || B || D_2)$

M3: $B \rightarrow A$: $\text{Cert}B || R_B || R_A || A || D_5 || S_B(R_B || R_A || A || D_4)$

The random number R_A is present in the signed part of **M2** to prevent *B* from obtaining the signature of *A* on data chosen by *B*. For a similar reason the random number R_B is present in the signed part of **M3**.

However, this approach cannot completely avoid the abuse of signatures for the following two reasons.

1. Although both *A*'s and *B*'s nonces are included in both *A*'s and *B*'s signatures, *B* selects its nonce before *A*. This means that *A* is possibly in a more favourable position than *B* to misuse the other party's signature. In order to prevent this, *B* can generate an unpredictable nonce and add the nonce into the message before signing it, e.g. in **M3** an extra unpredictable nonce can be included in D_4 and D_5, i.e. **M3** becomes:

$$B \rightarrow A: \ \text{Cert}B || R_B || R_A || A || R'_B || D'_5 || S_B(R_B || R_A || A || R'_B || D'_4)$$

2. It is possible for the users of the signatures to 'bypass' some nonces involved in the protocol if other signatures in different contexts use nonces in the same way. For example, a different protocol might make use of a message $S_A(R||X||B||D_2)$ or a message $S_B(R||Y||A||D_4)$, where *R* is a random number and *X* or *Y* has some particular meaning. The signatures of the previous protocol could then potentially be successfully abused in that protocol.

The above discussion implies that changing the protocol construction just makes abuse of digital signatures more difficult, and cannot protect against such attacks completely, because the protocol itself cannot detect the misuse of the digital signatures involved. However, there do exist means of avoiding these problems, such as the following.

- In [20] a method called 'key separation' is proposed, i.e. using different keys for different applications so as to avoid the misuse of digital signatures. Such a technique is well established, particularly in the context of symmetric cryptography.
- Another approach to avoid the abuse of signatures is using sequence numbers rather than unpredictable nonces to control the uniqueness of authentication exchanges. Because the values of the sequence numbers are agreed by both the claimant and verifier, neither party to a protocol can be persuaded to sign information which has some 'hidden meaning' (for further details of sequence numbers see Annex B of ISO/IEC 9798-4).
- Last, but not least, note that zero knowledge based protocols are specifically designed to prevent this type of attack.

4.3.7 Predictable and unpredictable nonces

Nonces are used as challenges to verify the freshness and timeliness of exchange messages in authentication protocols. Those protocols specified in Parts 2, 3 and 4 of ISO/IEC 9798 which are based on nonces, all require the nonces used to be unpredictable, i.e. the nonces must either be random numbers or pseudo-random numbers which are generated in such a way that intercepting third parties cannot guess future nonce values. However, in some circumstances it is advantageous to use nonces which are predictable (e.g. if they are generated using a counter or clock). For example, it may be difficult for an entity to generate random or unpredictable pseudo-random numbers.

We now consider how secure nonce-based authentication protocols can be devised even if nonces are predictable (as long as they are still 'one time'). This can be achieved by cryptographically protecting all the messages in a protocol including the nonces.

Before proceeding we briefly distinguish between predictable nonces and sequence numbers, both of which are used to control the uniqueness of authentication exchanges. They are both used only once within a valid period of time and can be predicted in advance by a third party. However, a predictable nonce is used as a challenge, so that the responder does not need to know it before receiving it and to record it after using it. A sequence number is agreed by the claimant and verifier beforehand according to some policy, and will be rejected if it is not in accordance with the agreed policy. Furthermore, use of sequence numbers may require additional 'book keeping' for both the claimant and verifier.

It has been observed in [21] that in an authentication protocol using symmetric encryption with a nonce, if the nonce is unpredictable then either the challenge or the response can be transmitted unencrypted; if the nonce is predictable both the challenge and response have to be transmitted encrypted. Otherwise the protocol cannot protect against replay attacks.

We now illustrate that this logic also applies to protocols using digital signatures and CCFs. One example of how digital signatures can be used in conjunction with a predictable nonce to produce a secure authentication protocol is the following, which is a modification of the protocol given in Clause 5.2.2 of ISO/IEC 9798-3.

M1: $B \rightarrow A$: $\mathrm{Cert}B||R_B||S_B(R_B)$

M2: $A \rightarrow B$: $\mathrm{Cert}A||R_A||S_A(R_A||R_B||B)$

M3: $B \rightarrow A$: $S_B(R_B||R_A||A)$

Note that the modification is to include a signed copy of the nonce in message **M1**, thereby preventing a 'so called' preplay attack.

Another example, this time based on the use of CCFs, is the following, which is a modification of the protocol given in Clause 5.2.2 of ISO/IEC 9798-4.

M1: $B \rightarrow A$: $R_B||F_K(R_B)$

M2: $A \rightarrow B$: $R_A||F_K(R_A||R_B||B)$

M3: $B \rightarrow A$: $F_K(R_B||R_A)$

The above analysis means that there is a good range of protocols available to support both unpredictable and predictable nonces.

Note that when using a predictable nonce as a challenge, since a future challenge is predictable to the responder, the verifier of the challenge has to depend on the honesty and competence of the responder, [21].

4.3.8 Disclosure of plaintext/ciphertext pairs

There are a number of different models for known plaintext attacks, chosen plaintext attacks and chosen ciphertext attacks on cipher systems (e.g. [22,23]). Although obtaining plaintext/ciphertext pairs far from guarantees that attacks will be successful, it is typically the necessary first step for an attacker. Whether or not an attacker has access to a plaintext/ciphertext pair during the authentication processing depends on both the authentication protocol and the details of the cryptographic processing.

We use the term 'plaintext/ciphertext pair' rather loosely here, to cover matching pairs of input and output for a variety of cryptographic algorithms, including encipherment algorithms, digital signature schemes and cryptographic check functions. The question as to whether disclosing a plaintext/ciphertext pair is a problem depends on two main things:

- the strength of the algorithm, and

- whether the same algorithm and key are used for applications other than the authentication exchange.

It is certainly the case that in some situations the disclosure of plaintext/ciphertext pairs is not a security problem and that is what the ISO/IEC 9798 protocols assume. However we are concerned here with situations where disclosure of pairs may be a security threat, and we consider ways in which the threat may be avoided.

The following examples of unilateral authentication protocols are described in ISO/IEC 9798 parts 2, 3 and 4 respectively.

Example 1: Symmetric encryption with nonce (ISO/IEC 9798-2):

M1: $B \rightarrow A: \; R_B || D_1$

M2: $A \rightarrow B: \; D_3 || E_{K_{AB}}(R_B || B || D_2)$

Example 2: Digital signature with timestamp (ISO/IEC 9798-3):

M1: $A \rightarrow B: \; T_A || B || D_1 || S_A(T_A || B || D_2)$

Example 3: CCF with sequence number (ISO/IEC 9798-4):

M1: $A \rightarrow B: \; N_A || B || D_1 || F_{K_{AB}}(N_A || B || D_2)$

Whether these protocols provide plaintext and ciphertext pairs depends on what kind of the optional text field D_2 is used. If D_2 is predictable data (including null), a plaintext/ciphertext pair is exposed. If D_2 includes some unpredictable data, whether the plaintext/ciphertext pair is exposed depends only on the structure of the cryptographic operation applied. This second case seems unsuitable for CCF-based message exchanges.

If D_2 is predictable data, in timestamp or sequence number based protocols, it is rather difficult to avoid disclosure of plaintext/ciphertext pairs, since the intruder can choose when to start the protocol or what predictable number to be used in the protocol. When using a random number or unpredictable pseudo-random number as a nonce, the nonce has to be cryptographically protected in order to let the protocols not disclose any known plaintext/ciphertext pair. The following examples of protocols which do not disclose plaintext/ciphertext pairs do not depend on whether or not D_1, D_2 and D_3 are unpredictable.

Example 4: Symmetric encryption with nonce:

M1: $B \rightarrow A: \; E_{K_{AB}}(R_B || D_1)$

M2: $A \rightarrow B: \; D_3 || E_{K_{AB}}(R_B || B || D_2)$

Example 5: Digital signature and asymmetric encipherment with nonce:

M1: $A \rightarrow B: \; E_B(R_A || B || D_1 || S_A(R_A || B || D_2))$

Example 6: CCF with nonces:

M1: $A \rightarrow B: \; R_A || B || D_1 || F_{K_{AB}}(R_A || B || D_2) \oplus R'_A || F_{K_{AB}}(R'_A || D_3)$

Note that the first nonce R_A in Example 6 can be replaced by a timestamp T_A or sequence number N_A.

4.3.9 Using poorly synchronised clocks

The authentication protocols with timestamps specified in ISO/IEC 9798 require the communicating parties to have synchronised clocks, and they depend on them to prevent valid exchanged messages from being replayed at a later time. There are several approaches to achieving secure clock synchronisation and re-synchronisation (e.g. [24,25,26]). However, in some environments, time

stamp based authentication protocols need to be used although the parties do not have exactly synchronised clocks. For example, in a mobile telecommunications system, a mobile user may find it difficult always to keep a clock synchronised with the clock of its service provider. The user and network may still wish to use a timestamp-based entity authentication protocol rather than a nonce-based one to reduce the number of messages exchanged, and to allow the detection of forced delays.

Two points must be considered when using a timestamp-based protocol in such an environment. Firstly the size of the acceptance window to match the poorly synchronised clocks must be selected. The size of this window can be either fixed or dynamically changed depending on the environment in which the authentication protocol is used, in particular on the delays in the communications channels and the quality of the clocks in use. Secondly all messages within the current acceptance window must be logged, and second and subsequent occurrences of identical messages within that window must be rejected (see Annex B of ISO/IEC 9798-2 and [27]).

There are a variety of possible attacks whose viability depends on the nature of the communications (see subsection 4.3.3), e.g. a suppress-replay attack can happen in a non-authenticated channel, and a classic replay attack can happen in an equal rights channel or a non-authenticated channel. The use of timestamps are with respect to different communications channels. For example, when the sender's clock is significantly ahead of the receiver's clock, and the message with the sender's timestamp is received in time, it can be rejected because the receiver can find that it is a post-dated message.

Furthermore, this post-dated message can protected against a classic replay attack by using the previous approach. However, this post-dated message can be attacked by a suppress-replay attack (see [28]). In this case, the protocol may remain vulnerable even after the faulty clock has been resynchronised because that post-dated message could be replayed in a corrected time later. One instance of a suppress-replay attack is where a bogus base station in a mobile telecommunications system, which may be closer to a mobile user than a valid base station, could suppress the user's messages and replay them later.

4.3.10 Summary

We have seen how the properties of underlying mechanisms affect the design of authentication protocols and how to tailor authentication protocols to match the underlying mechanisms available. The properties of the mechanisms are induced by the particular environments in which the protocol works, and by the particular resources of the authenticating entities. A number of alternatives to the ISO/IEC 9798 protocols, which do not make such strict requirements on the underlying mechanisms, have been proposed and analysed. We now summarise them as follows.

- If individual authentication servers are untrustworthy for clients, three possible approaches are: (1) allowing clients to choose trustworthy servers, (2) using multiple authentication servers instead of a single one, (3) using off-line master authentication servers.

- In order to preserve entity identity privacy, two possible methods are: (1) based on asymmetric cryptography the entity identity can be hidden by using the public part of the receiver's asymmetric key pair, (2) one or more temporary entity identities instead of the real entity identity can be transmitted unencrypted.

- In order to avoid abuse of digital signatures, four possible approaches are: (1) let the signer generate an unpredictable nonce and insert the nonce into the message before signing it, (2) use different keys for different applications, (3) use sequence numbers rather than unpredictable nonces to control the uniqueness of authentication exchanges, (4) use zero knowledge techniques.

- When using a predictable nonce as a challenge, all messages, both the challenge and response, have to be protected cryptographically. Some possible examples are: (1) in a symmetric encryption based protocol, the challenge and response are respectively a nonce and a function of the nonce encrypted by the shared key, (2) in a digital signature based protocol, the private parts of the challenger's and responder's asymmetric key pairs are used in the challenge and response respectively, (3) in a CCF based protocol, the challenge is a nonce concatenated with a CCF of the nonce, and the response is a CCF of a function of the nonce.

- In unpredictable nonce based protocols without unpredictable optional text fields it is possible to avoid disclosing plaintext/ciphertext pairs by using cryptographic protection for every message. In timestamp or sequence number based protocols without unpredictable optional text fields, it appears to be rather difficult to avoid giving plaintext/ciphertext pairs.
- When using poorly synchronised clocks in authentication protocols, one approach is to take the following steps: (1) select the size of the acceptance window to match the poorly synchronised clocks, (2) log all messages and reject the second and subsequent occurrences of identical messages within the acceptance window.

4.4 Authentication and key distribution using multiple servers

The text in this section is based on two papers, [29,30], prepared for publication as part of the 3GS3 project.

Some recent research on key distribution systems has focused on analysing trust in authentication servers, and constructing key distribution protocols which operate using a number of authentication servers, which have the property that a minority of them may be untrustworthy. We propose here two key distribution protocols with multiple authentication servers using a cross checksum scheme. Both protocols are based on the use of symmetric encryption for verifying the origin and integrity of messages.

In these protocols it is not necessary for clients to trust an individual authentication server. A minority of malicious and colluding servers cannot compromise security and their inappropriate behaviour can be detected. The first 'parallel' protocol can prevent a minority of servers disrupting the service. The second 'cascade' protocol has to work with other security mechanisms in order to prevent a server breaking the procedure by refusing to cooperate. As compared with other proposed protocols with similar properties these two protocols require fewer exchanged messages and less computational effort.

4.4.1 Introduction

In the context of symmetric cryptography, if two entities sharing no secret want to securely communicate with each other, they typically do so with the assistance of a third party. Typically this third party is an authentication server who provides an authentication service including distributing a secure session key to these entities as clients. Such an authentication server is sometimes referred to as a trusted third party since every client has to trust it by sharing a secret with it. The security of a typical key distribution protocol depends on the assumption that the authentication server is trustworthy. Unfortunately not every authentication server is always trustworthy. If the authentication server is malicious, or is compromised, the security of communications between these clients cannot be guaranteed.

In order to make a protocol work in an environment where clients do not trust an individual server, it is important to find authentication schemes which reduce the requirement for trusting servers. Some recent research [17,18,31,32,33] has focused on analysing trust in authentication servers, and constructing secure key distribution protocols which do not require trusting individual authentication servers. As has already been described in subsection 4.3.4, a range of possible approaches exist where the use of a more complex authentication service results in a more secure and available authentication service. We concentrate here on the second of the three approaches described in subsection 4.3.4, namely where many servers are used simultaneously to achieve authentication.

Gong, [17], proposed a protocol with multiple authentication servers such that a minority of malicious and colluding servers cannot compromise security or disrupt service. In that protocol two clients participate in choosing a session key, and each relevant server is responsible for converting and distributing a part of the session key. However in some environments it is necessary to let servers, not clients, choose a session key, for example, in environments where clients cannot randomly and securely generate a session key or candidate keys. One potential problem with letting servers be involved in choosing candidate session keys is providing clients with the means to verify that they are provided with good candidate keys. In this subsection, a candidate session key is 'good' if it is received by both clients. In order to solve this problem, we discuss a cross checksum scheme, which works under the assumption that more than half the servers follow the protocol specifications correctly. We let all servers participate in choosing candidate session keys.

Each server randomly and securely generates a candidate key. Two clients participate in verifying these keys by exchanging the relevant check values using a one-way globally known hash function in order to check whether every candidate key is received by both clients correctly. These clients then eliminate bad candidate keys and use good keys to compute a final session key. It is not necessary for clients to trust any individual server because no single server can know the session key.

Based on this scheme we propose two key distribution protocols with an arbitrary number of authentication servers. In these protocols a minority of malicious and colluding authentication servers cannot compromise security and can be detected. The first 'parallel' protocol can prevent a minority of servers disrupting the service. The second 'cascade' protocol has to work with other security mechanisms in order to prevent a server breaking the procedure by refusing to cooperate. If more than half the servers follow the protocol specifications correctly, the following properties will hold for the session key:

- it will be fresh (i.e. not a replay of an old key) and random (i.e. not predictable by any party),
- it will be known only to the clients and not to any server,
- it can be checked by both clients, and
- it has not been chosen by any individual client or server.

The two protocols with n servers we discuss here use $2n+4$ and $n+5$ messages, whereas Gong's protocol, [17], uses $4n+3$ messages.

The rest of this discussion is organised as follows. The relevant notation and assumptions made in this subsection are described immediately below. A typical authentication protocol and how trust problems can arise are discussed in subsection 4.4.3. We then briefly illustrate Gong's protocol, [17], in subsection 4.4.4. After that we analyse a cross checksum scheme in subsection 4.4.5. Based on this scheme we propose two key distribution protocols, one of which is a parallel protocol specified in subsection 4.4.6 and the other is the cascade protocol in subsection 4.4.7. We conclude in subsection 4.4.8.

4.4.2 Notation and assumptions

We now review the main assumptions made here. All protocols considered here are based on symmetric encipherment. We assume that two entities A and B wish to communicate securely with each other. For this purpose they need to verify each others' identities and to establish a shared session key, but before the authentication processing starts they do not currently share any secret. So they make use of a third party, an authentication server S (or a set of servers $S_1,S_2,...,S_n$).

In the protocols using a multiplicity of servers we also assume that:

- A and B do not trust any individual server,
- A and B believe that more than half the servers will correctly follow the protocol specifications,
- A and B share secret keys K_{AS_i} and K_{BS_i} respectively with S_i.

We assume that the encipherment operation used provides origin authentication and integrity services, i.e. a received message encrypted using a shared secret must have been sent by the possessor of the secret in the form that it was received.

In the protocol descriptions,

- $A \rightarrow B$: m indicates that A sends message m to B,
- $\{m\}_K$ denotes m encrypted with the key K,
- $x||y$ denotes the concatenation of data items x and y,
- g and h are one-way hash functions,
- $\lfloor m \rfloor$ denotes the integer part of m, and
- $\lceil m \rceil$ denotes the smallest integer not less than m.

4.4.3 A typical protocol and an associated problem

We now discuss a typical protocol which uses symmetric encryption techniques and lets an authentication server *S* choose the session key.

This protocol is similar to a protocol in Clause 6.3 of ISO/IEC DIS 11770-2, [5]. In this protocol we suppose that *A* and *S* share a secret key K_{AS}, and *B* and *S* share another secret key K_{BS}.

M1: $A \rightarrow B: \; A||B||R_A$

M2: $B \rightarrow S: \; A||B||R_A||R_B$

M3: $S \rightarrow B: \; \{A||R_B||K_{AB}\}_{K_{BS}}||\{B||R_A||K_{AB}\}_{K_{AS}}$

M4: $B \rightarrow A: \; \{B||R_A||K_{AB}\}_{K_{AS}}||\{A||R_A||R'_B\}_{K_{AB}}$

M5: $A \rightarrow B: \; \{B||R'_B||R_A\}_{K_{AB}}$

where R_A, R_B and R'_B are three nonces, and K_{AB} is a session key for *A* and *B*.

A sends *B* a message **M1** containing a nonce R_A as a challenge; *B* then sends *S* a message **M2** including nonces R_A and R_B; *S* chooses a session key K_{AB} and distributes it to *A* and *B* in **M3**; by checking the enclosed nonces, *A* and *B* verify that the reply of *S* is fresh and retrieve K_{AB}; finally *A* and *B* complete a handshake. After this protocol has executed, *A* and *B* have agreed upon session key K_{AB}, and *A* and *B* believe that K_{AB} has been retrieved by each other and is appropriate for use between them.

A possible problem with this protocol is that *A* and *B* have to trust *S* in order to obtain the authentication information and the session key. *S* keeps total control over communications between *A* and *B* because *S* does all the 'verification of identities' and knows the session key. This protocol is vulnerable to active 'untrustworthy third party' attacks. That is, if *S* is malicious or corrupted, it can intercept communications between *A* and *B*, can impersonate *A* to *B* or *B* to *A*, and can leak *A*'s and *B*'s secrets.

4.4.4 Gong's protocol with multiple servers

In order to solve this problem with the above protocol, Gong, [17], proposed a different protocol, again based on symmetric encryption, but with *n* servers $S_1, S_2, ..., S_n$ instead of one server. Each server is responsible for converting and distributing a part of the session key. In the following protocol, the steps **M1**$_i$, **M2**$_i$, **M4**$_i$ and **M5**$_i$ are repeated for $i = 1,...,n$. Steps **M1**$_i$ and **M2**$_i$ must all be completed before step **M3** is performed; similarly steps **M4**$_i$ and **M5**$_i$ must all completed before step **M6**.

M1$_i$: $A \rightarrow S_i: \; A||B$

M2$_i$: $S_i \rightarrow A: \; R_{S_i}$

M3: $A \rightarrow B: \; A||B||R_A||R_{S_1}||\{A||B||R_{S_1}||x_1||C(x)\}_{K_{AS_1}}||...||R_{S_n}||\{A||B||R_{S_n}||x_n||C(x)\}_{K_{AS_n}}$

M4$_i$: $B \rightarrow S_i: \; A||B||R_A||R_B||\{A||B||R_{S_i}||x_i||C(x)\}_{K_{AS_i}}||\{B||A||R_{S_i}||y_i||C(y)\}_{K_{BS_i}}$

M5$_i$: $S_i \rightarrow B: \; \{B||R_A||y_i||C(y)\}_{K_{AS_i}}||\{A||R_B||x_i||C(x)\}_{K_{BS_i}}$

M6: $B \rightarrow A: \; \{B||R_A||y_1||C(y)\}_{K_{AS_1}}||...||\{B||R_A||y_n||C(y)\}_{K_{AS_n}}||\{R_A\}_{K_{AB}}||R_B$

M7: $A \rightarrow B: \; \{R_B\}_{K_{AB}}$

A (or *B*) chooses a candidate session key *x* (or *y*) and computes $x_i = f_{t,n}(x,i)$ (or $y_i = f_{t,n}(y,i)$) for each server S_i. Here $f_{t,n}$ is a threshold function, [34], that produces *n* shadows of *x* (or *y*) in such a way that it is easy to recover *x* (or *y*) from any *t* shadows, but less than *t* shadows reveal no information about *x* (or *y*). To compute $f_{t,n}(x)$ (or $f_{t,n}(y)$), *A* (or *B*) chooses a random polynomial $p(x)$ of degree *t*-1 (or $p(y)$) with $p(0) = x$ (or $p(0) = y$). *A* (or *B*) then computes $x_i = f_{t,n}(x,i) = p(i)$ (or $y_i = f_{t,n}(y,i) = p(i)$), i=1,...,*n*. Due to the property of interpolation, given any *t* of the x_i's (or y_i's), *B* (or *A*) can easily determine $p(x)$ (or $p(y)$) and recover $x = p(0)$ (or $y = p(0)$), [35]. With less than *t* shadows, no information about *x* (or *y*) can be determined.

In this protocol, a cross checksum scheme is used to verify the legitimacy of the shadows. Cross checksums for *x* and *y* are defined as $C(x) = (g(x_1),...,g(x_n))$ and $C(y) = (g(y_1),...,g(y_n))$, respectively, where *g* is a one-way hash function. The session key is computed by $K_{AB} = h(x,y)$, where *h* is a pre-determined one-way hash function. In order to prevent *A* or *B* imposing the session key, the choice of *h* is limited; for example, it cannot be an exclusive-or operation. In this protocol, the total number of messages is $4n+3$.

It was said in Gong's paper that in fact there is a major difficulty in letting servers be involved in choosing the session key, because clients do not have a secure communication channel for verification purposes before authentication completes, and thus they cannot easily reach an agreement on what they have received from which servers. As mentioned in subsection 4.4.1, in some environments it is useful to let servers, not clients, choose a session key. We now propose a cross checksum scheme which is the basis for the two new protocols given below. Using this cross checksum scheme, all servers can participate in choosing candidate session keys, and two clients are able to verify them and use all good candidate keys to compute a session key.

4.4.5 Cross checksums of candidate keys

Cross checksum schemes were used by Gong, [17],and Klein et al., [32], in key distribution protocols. This section discusses a particular cross checksum scheme which can be used for authenticated key establishment. We ignore for the moment the issue of verifying the 'freshness' of the keys, i.e. we ignore the entity authentication issues. In the next two sections we describe authentication protocols which use this cross checksum technique, and thus provide verified session key establishment between *A* and *B*.

The cross checksum scheme works as follows.

Algorithm 4.1

1. S_i generates candidate session keys K_{Ai} and K_{Bi}, and sends them encrypted under K_{ASi} and K_{BSi} to *A* and *B* respectively. If K_{Ai} and K_{Bi} are good candidate keys, then they satisfy $K_{Ai} = K_{Bi}$.

2. *B* computes the check values $g(K_{Bi})$ of the candidate keys K_{Bi}, where g is a globally known one-way hash function, then performs the following calculations and sends $G'_B(1),...,G'_B(n)$ to *A*.

$$g'(K_{Bi}) = \begin{cases} g(K_{Bi}) & \text{if } B \text{ believes that it has received the candidate key} \\ EM1 & \text{otherwise} \end{cases}$$

$$G_B = g'(K_{B1})||...||g'(K_{Bn})$$

$$G'_B(i) = \begin{cases} \{G_B\}_{K_{Bi}} & \text{if } B \text{ believes that it has received the candidate key} \\ EM2 & \text{otherwise} \end{cases}$$

where EM1 and EM2 are error messages.

3. On receipt of $G'_B(1),...,G'_B(n)$ from B, A computes the check values $g(K_{Ai})$ of the candidate keys K_{Ai}, then performs the following calculations and sends $G'_A(1),...,G'_A(n)$ to B.

$$g'(K_{Ai}) = \begin{cases} g(K_{Ai}) & \text{if } A \text{ believes that } A \text{ and } B \text{ have received the same key} \\ EM1 & \text{otherwise} \end{cases}$$

$$G_A = g'(K_{A1})||...||g'(K_{An})$$

$$G'_A(i) = \begin{cases} \{G_A\}_{K_{Ai}} & \text{if } A \text{ believes that } A \text{ and } B \text{ have received the same key} \\ EM2 & \text{otherwise} \end{cases}$$

A possible set of message exchanges is then as follows, where the first step **M1*i*** is repeated for i=1,...,n. Steps **M1*i*** must all be completed before step **M2**.

M1*i*: $S_i \rightarrow B$: $\{K_{Bi}\}_{K_{BS_i}}||\{K_{Ai}\}_{K_{AS_i}}$

M2: $B \rightarrow A$: $\{K_{A1}\}_{K_{AS_1}}||...||\{K_{An}\}_{K_{AS_n}}||G'_B(1)||...||G'_B(n)$

M3: $A \rightarrow B$: $G'_A(1)||...||G'_A(n)$

We now show how A and B can use the exchanged information to agree a session key. In order for this process to work, A and B must trust at least $\lfloor n/2+1 \rfloor$ of the servers to behave correctly, i.e. at least $\lfloor n/2+1 \rfloor$ of the pairs (K_{Ai},K_{Bi}) satisfy $K_{Ai} = K_{Bi}$.

On receipt of **M1*i*** (or waiting a time-out period if an expected message does not arrive), B firstly checks whether the message with a correct syntax from every server has been received, and then creates a sequence $G_B = g'(K_{B1}),...,g'(K_{Bn})$. Because g is a one-way hash function it does not disclose the secret of the corresponding candidate key. B now wants to show G_B to A. However it is necessary to guarantee that any server can read G_B but cannot modify it without being detected. One solution is for B to send A another sequence $G'_B(1),...,G'_B(n)$.

After receiving **M2**, A generates a similar sequence $G'_A(1),...,G'_A(n)$ by checking whether both A and B have received the same key. A firstly decrypts each $G'_B(i)$ using K_{Ai}. Because more than half the servers follow the protocol specifications correctly, some K_{Ai} may not be the same as K_{Bi}, but at least $\lfloor n/2+1 \rfloor$ of the values K_{Ai} are equal to the corresponding values K_{Bi}; A may not get all G_B encrypted by B, but A can decrypt at least $\lfloor n/2+1 \rfloor$ G_B copies; some copies may not be the same as others, but at least $\lfloor n/2+1 \rfloor$ copies must be same. Checking these copies, A eliminates the minority of different copies and keeps the majority, so long as the majority contains > $n/2$ elements. A then computes the values of $g(K_{Ai})$, and compares them with the values of $g(K_{Bi})$ included in G_B. After that A creates G_A and $G'_A(1),...,G'_A(n)$.

On receipt of **M3**, B checks it in the same way. Finally A and B retrieve all 'good' candidate keys which are used to construct a session key K_{AB} = h(all the good candidate keys), where h is a pre-determined one-way hash function.

Theorem 4.2

Using the above cross checksum scheme, A and B can retrieve all good candidate session keys, and the number of these keys is at least $\lfloor n/2+1 \rfloor$, given the following assumptions:

1. There are m servers out of a total n servers following the protocol specifications correctly, where

$$m \geq \lfloor n/2+1 \rfloor.$$

2. A and B both correctly follow the protocol specifications.

3. The *n-m* bad servers can possibly do the following:

    - not send messages to *A* and *B* or send messages with the wrong syntax;
    - send different candidate keys to *A* and *B*;
    - eavesdrop on $G'_A(1),...,G'_A(n)$ and $G'_B(1),...,G'_B(n)$; and
    - modify $G'_A(1),...,G'_A(n)$ or $G'_B(1),...,G'_B(n)$ in transit between *A* and *B*.

Note that the system will clearly fail to establish a session key if malicious entities (either dishonest servers or other third parties) interfere with communications between *A* and *B* (or between *B* and the honest servers). We therefore assume that *A* and *B* will request messages to be sent again until the protocol succeeds.

Proof

Suppose that *m* is the number of 'good' servers, where $m \geq \lfloor n/2+1 \rfloor$, and *j* is the number of (K_{Ai},K_{Bi}) pairs which are received by *A* and *B*, and satisfy $K_{Ai} = K_{Bi}$.

1. On the above assumptions and the property of encipherment algorithms assumed in subsection 4.4.2, *j* must satisfy $j \geq m \geq \lfloor n/2+1 \rfloor > n/2$.
2. If $G'_B(1),...,G'_B(n)$ have not been modified by bad servers, *A* obtains *j* copies of G_B encrypted by *B*. A combination of the *n-j* malicious servers can only construct at most *n-j* consistent values of an 'incorrect' G_B, because of the use of encryption, and $n\text{-}j \leq n\text{-}\lfloor n/2+1 \rfloor = \lceil n/2-1 \rceil < n/2$.
3. Hence *A* retrieves $j > n/2$ values K_{Ai} which are also received by *B*. Furthermore G_A includes *j* check values and *n-j* error messages EM1, and $G'_A(1),...,G'_A(n)$ includes *j* copies of G_A encrypted by *A* and *n-j* error messages EM2.
4. For the same reasons mentioned in items 2 and 3, *B* obtains G_A generated by *A* and retrieves *j* copies of K_{Bi} which also are retrieved by *A*.

We arrive at the conclusion that *A* and *B* can obtain $> n/2$ pairs (K_{Ai},K_{Bi}) which satisfy $K_{Ai} = K_{bi}$. These values can then be used to compute a secret session key K_{AB}.

Theorem 4.3

The above cross checksum scheme allows a group of collaborating servers to learn the session key (or force *A* and *B* to agree on different session keys) if $m \leq \lceil n/2-1 \rceil$. If $\lceil n/2-1 \rceil < m < \lfloor n/2+1 \rfloor$ (i.e. $m = n/2$) then colluding servers can always prevent *A* and *B* agreeing on a session key.

Proof

We first suppose that $m \leq \lceil n/2-1 \rceil$, and hence $n\text{-}m \geq \lfloor n/2+1 \rfloor > n/2$.

Suppose that the *m* 'good' servers are $S_1,...,S_m$, and the *n-m* 'bad' servers are $S_{m+1},...,S_n$ (and that the 'bad' servers are all colluding). These *n* servers send *n* pairs (K_{Ai},K_{Bi}) to *A* and *B* (which may not agree). *B* creates $G'_B(1),...,G'_B(n)$ where

$$G'_B(i) = \{g(K_{B1})||...||g(K_{Bn})\}_{K_{Bi}}, \qquad (1 \leq i \leq n).$$

The *n-m* 'bad' servers then modify $G'_B(1),...,G'_B(n)$ to $\Gamma'_B(1),... \Gamma'_B(n)$, where

$$\Gamma'_B(i) = \begin{cases} G'_B(i) & \text{if } 1 \leq i \leq m \\ \{\text{EM1}||...||\text{EM1}||g(K_{A(m+1)})||...||g(K_{An})\}_{K_{Bi}} & \text{if } m+1 \leq i \leq n \end{cases}$$

There are two simple ways in which $S_{m+1},...,S_n$ can invalidate the protocol with this strategy. Firstly, if they follow the protocol correctly and put $K_{Ai} = K_{Bi}$ $(m < i \leq n)$, then, on receipt of

$\Gamma'_B(1),...\ \Gamma'_B(n)$, *A* will deduce that the first *m* candidate keys are bad and the last *n-m* candidate keys are good (this will follow since $n - m > n/2$). *A* will then eliminate the 'bad keys' and keep the 'good keys' in $G'_A(1),...,G'_A(n)$, where

$$G'_A(i) = \begin{cases} \text{EM2} & \text{if } 1 \le i \le m \\ \{\text{EM1}||...||\text{EM1}||g(K_{A(m+1)})||...||g(K_{An})\}_{K_{Ai}} & \text{if } m+1 \le i \le n \end{cases}$$

After receiving $G'_A(1),...,G'_A(n)$, *B* believes that *A* has retrieved the same last *n-m* candidate keys. *A* and *B* use these candidate keys to compute the final session key, which means that the *n-m* colluding servers can obtain the session key.

Secondly, the 'bad servers' can choose $K_{Ai} \neq K_{Bi}$ ($m < i \le n$). By changing K_{Bi} to K_{Ai} in the appropriate places in the message from *B* to *A* (as above), and then changing them back in the return message, the 'bad servers' can force *A* and *B* into the situation where they believe they have agreed on a key, but in fact they have different keys; to make matters even worse both the key held by *A* and the key held by *B* will be known to the colluding 'bad servers'.

Put simply, in both attacks the majority of 'bad' servers can shut out the minority of 'good' servers and either make *A* and *B* agree on different keys and/or make them agree on a key known to all the 'bad' servers.

Finally, if $\lceil n/2-1 \rceil < m < \lfloor n/2+1 \rfloor$ (i.e. $m = n/2$), then, by failing to follow the protocol correctly, the 'bad servers' can prevent the protocol ever operating correctly, even if they do not interfere with communications between *A* and *B* (or between *A*/*B* and the 'good' servers), by preventing a majority of candidate keys ever being established.

4.4.6 A parallel protocol

In this section we present a key distribution protocol based on the cross checksum method of the previous section. *A* initiates the protocol by sending a request to *B* in **M1**.

M1: $A \rightarrow B: \ A||B||R_A$

B then contacts every server S_i to let him choose a candidate session key K_i. The following steps $\mathbf{M2}_i$ and $\mathbf{M3}_i$ are repeated for $i = 1,...,n$:

$\mathbf{M2}_i$: $B \rightarrow S_i: \ A||B||R_A||R_B$

$\mathbf{M3}_i$: $S_i \rightarrow B: \ \{A||R_B||K_i\}_{K_{BS_i}}||\{B||R_A||K_i\}_{K_{AS_i}}$

After receiving answers from the *n* servers (or waiting a time-out period if an expected message does not arrive), *B* organises two sequences $G_B = g'(K_1),...,g'(K_n)$ and $G'_B(1),...,G'_B(n)$, as defined in Algorithm 1, and then sends the following message to *A*:

M4: $B \rightarrow A: \ \{B||R_A||K_1\}_{K_{AS_1}}||...||\{B||R_A||K_n\}_{K_{AS_n}}||G'_B(1)||...||G'_B(n)$

On receipt of **M4**, *A* checks it and organises two similar sequences $G_A = g'(K_1),...,g'(K_n)$ and $G'_A(1),...,G'_A(n)$, as specified in Algorithm 1. *A* then sends $G'_A(1),...,G'_A(n)$ to *B* in **M5**. *B* checks it in the same way. *A* and *B* thus obtain the same good candidate keys to construct a session key $K_{AB} = h$(all the good candidate keys). The last two steps are a handshake to inform each other that the correct session key has been retrieved.

M5: $A \rightarrow B: \ G'_A(1)||...||G'_A(n)||\{B||R_B||R'_A\}_{K_{AB}}$

M6: $B \rightarrow A: \ \{A||R'_A||R_B\}_{K_{AB}}$

In this protocol, if at least one good K_i is random and fresh it is guaranteed that the session key K_{AB} is random and fresh. Because only *A* and *B* know all the good candidate session keys, the session key K_{AB} is known only to *A* and *B* and not to any server (as long as $n \geq 2$). Since K_{AB} results from the verification between *A* and *B*, it is verifiable for both *A* and *B*. No one among the *n* servers and the two clients can impose K_{AB}. So the session key K_{AB} in this protocol satisfies the four properties mentioned in subsection 4.4.1 on the assumption that more than half the servers follow the protocol specifications correctly.

As compared with Gong's protocol with similar properties, this protocol has the following advantages.

1. The number of messages in this protocol is $2n+4$ which is less than $4n+3$.
2. Because the candidate session keys are chosen by servers, and no individual client can 'impose' the resultant session key, the choice of *h* is less limited than in Gong's protocol; for instance, an exclusive-or operation can be used here, which cannot be used in Gong's protocol.
3. The computational complexity of this protocol is lower because no polynomial interpolation is used.

A possible disadvantage of this protocol is the potential length of messages **M4** and **M5**. G_B (or G_A) will contain *nt* bits (given *g* outputs a *t*-bit value) and hence $G'_B(i)$ (or $G'_A(i)$) will contain at least this number of bits. Hence **M4** and **M5** will contain $> n^2t$ bits. However, for practical applications, a typical choice for *t* might be 200 and *n* might be 20. This gives a message length of approximately 10 kbytes, which is not a particularly large value. Moreover, the total size of messages here is less than in Gong's protocol. The reason is that the size of G_A (G_B) is comparable with the size of $C(x)$ ($C(y)$), so the size of $G'_A(1)||...||G'_A(n)$ ($G'_B(1)||...||G'_B(n)$) is comparable with

$$\{C(x)\}_{K_{AS_1}}||...||\{C(x)\}_{K_{AS_n}} \quad (\{C(y)\}_{K_{BS_1}}||...||\{C(y)\}_{K_{BS_n}}).$$

In this protocol, such messages have to be transmitted once; however, in Gong's protocol, such messages are transmitted three times.

4.4.7 A cascade protocol

In this section we consider a second 'cascade' key distribution protocol again based on the cross checksum scheme of subsection 4.4.5. Note that this protocol works on the assumption that no server refuses to provide a service. The protocol is as follows:

M1: $A \rightarrow B$: $A||B||R_A$

M2: $B \rightarrow S_1$: $A||B||R_A||R_B$

The following step is repeated for $1 \leq i < n$:

M(*i*+2): $S_i \rightarrow S_{i+1}$:

$$A||B||R_A||R_B||\{A||R_B||K_1\}_{K_{BS_1}}||\{B||R_A||K_1\}_{K_{AS_1}}||...||\{A||R_B||K_i\}_{K_{BS_i}}||\{B||R_A||K_i\}_{K_{AS_i}}$$

The last four steps are as follows:

M(*n*+2): $S_n \rightarrow B$:

$$R_B||\{A||R_B||K_1\}_{K_{BS_1}}||\{B||R_A||K_1\}_{K_{AS_1}}||...||\{A||R_B||K_n\}_{K_{BS_n}}||\{B||R_A||K_n\}_{K_{AS_n}}$$

M(*n*+3): $B \rightarrow A$: $\{B||R_A||K_1\}_{K_{AS_1}}||...||\{B||R_A||K_n\}_{K_{AS_n}}||G'_B(1)||...||G'_B(n)$

M(*n*+4): $A \rightarrow B$: $G'_A(1)||...||G'_A(n)||\{B||R_B||R'_A\}_{K_{AB}}$

M(*n*+5): $B \rightarrow A$: $\{A||R'_A||R_B\}_{K_{AB}}$

The authentication request generated by *A* and *B* is sent to *n* servers via a cascade chain from S_1 to S_n. Every server randomly and securely chooses a candidate session key which is sent to *A* and *B* via the same cascade chain. Using an analysis similar to that in the previous section we can show that *A* and *B* can detect and eliminate every bad candidate session key which is not received by *A* and *B* correctly. *A* and *B* compute K_{AB} by using all the good candidate keys. Then *A* and *B* complete a two way handshake.

As in the previous analysis, in this protocol the session key K_{AB} also satisfies the four properties mentioned in subsection 4.4.1, on the assumption that more than half the servers follow the protocol specifications correctly. That means this protocol is as secure as the previous protocol. A major advantage of the protocol is that the number of messages is only n+5. Another advantage is that the clients do not need to signal back to every server. However the disadvantage of the protocol is that it is possible for a server to 'break' the procedure either maliciously or by mistake. For example, if a server simply refuses to cooperate, authentication and key distribution cannot be completed. One solution is to use this protocol with a control mechanism which can detect any server refusing to provide service, thus ensuring that no one can break the procedure.

4.4.8 Conclusions

Key distribution protocols without the assumption of trusting an individual authentication server are needed in some environments where clients have no reason to trust individual servers. A cross checksum scheme for the verification of candidate keys has been analysed. These candidate keys are generated by multiple servers in which no individual server is trusted but more than half of them are believed to behave correctly. Two protocols with an arbitrary number of authentication servers using the cross checksum scheme have been proposed.

On the assumption that more than half the servers follow the protocol specifications correctly, four desirable properties about the session key are guaranteed. The session key is: (1) random and fresh, (2) known only to clients and not to any server, (3) verifiable for every client, and (4) not imposed by any individual client or server.

A minority of malicious and colluding servers cannot compromise security in either protocol, and cannot break the procedure in the first protocol. The computational complexity, the number of exchanged messages and the size of total messages in the first protocol are lower than the protocol proposed by Gong, [17], with the same highly secure and available properties. The computational complexity and message number of the second protocol are lower than the first, with the same property of high security.

4.5 Authentication using minimally trusted servers

The text in this section is based on a paper, [36], prepared for publication as part of the 3GS3 project.

In this part of TR2 we consider further key distribution protocols using multiple authentication servers. The problem of key distribution using minimally trusted multiple servers is considered, and a new multiple server protocol is presented. In this protocol, if at least one server follows the protocol specifications correctly, a pair of clients can establish a shared session key, which cannot be learnt by any server. So long as not all the servers are colluding, no matter whether there is a server following the protocol correctly, no server can force any client to accept a wrong message.

4.5.1 Introduction

In this part of TR2 we consider another variation of the multiple authentication server problem previously considered in subsection 4.4.

We start in subsection 4.5.2 by discussing possible trust relationships between clients and servers during authentication and key distribution. Following this discussion we present an authentication and key distribution protocol using minimally trusted servers in subsection 4.5.3. The new protocol is based on a 3-party authentication and key distribution protocol recently proposed by Steiner et al. [37]. Steiner et al.'s protocol, using Diffie-Hellman key agreement [98], requires the third party to be trustworthy, because a dishonest server can subvert the protocol by impersonating one client to another. However, their protocol has the interesting feature that the resultant session key, although contributed to and 'strengthened' by the server, is not disclosed to him. In the new protocol, clients do not need to trust either an individual server or a majority of the servers.

In the subsequent four subsections, we discuss some interesting properties of this protocol. In subsection 4.5.4, we analyse security properties of the protocol. If at least one server follows the protocol specifications correctly, two clients can establish a shared session key, which cannot be learnt by any server. So long as not all the servers are colluding, no matter whether there is a server following the protocol correctly, no server can force any client to accept a wrong message. Once a session key is agreed by a pair of clients, no malicious entities (e.g. dishonest servers or other third parties) can discover it, even if any symmetric key shared between one client and one server is compromised. In subsection 4.5.5, we consider certain special trust relationships for which the protocol processing could be relaxed. In subsection 4.5.6, we consider the case where dishonest servers might have higher rights than clients to access the communications channels used to transmit message exchanges during the protocol processing, and where the protocol should add more stringent checks to prevent dishonest servers benefitting by suppressing exchanged messages. In subsection 4.5.7, we give an example of how to let multiple servers strengthen a session key in order to prevent one client from imposing the key.

The final subsection contains our conclusions.

4.5.2 Trust relationships

Simmons and Meadows [38] pointed out that an information integrity protocol involves a transfer of trust. For example, in a typical 3-party key distribution protocol, the two clients both share secret keys with a server. Each client trusts that the server will protect its key, and authenticate all client requests and responses. That means both clients believe the server will be honest and competent to follow the protocol specifications correctly. After the protocol succeeds, a session key is established between these two clients. So trust in the server, and in the integrity of the client's secure communication links with the server, has been transferred to trust in the integrity of the communication link, established using the session key, between two clients who had no prior trusted contact. If the server cannot be trusted then neither can the newly established link between clients.

As was mentioned earlier, the two clients may not trust an individual server, and so they might make use of a set of servers to complete identity verification and session key establishment. We now observe what kinds of trust could be transferred. In general, there are four different cases to consider, depending on how many servers are available and how many of them are trustworthy.

1. *Two clients trust a single authentication server*. This is the 'standard' case we have mentioned above.

2. *Two clients trust different authentication servers*. An example is provided by a mobile telecommunications system, where two communicating mobile users, who are registered with different service providers, only trust their own service providers. These two service providers collaborate to provide an authentication service to the users.

3. *Two clients trust some authentication servers from a set of such servers*. In this case, the clients do not trust any individual server, but they believe that some of the servers, which may be either a majority or a minority, will follow the protocol specifications correctly. One example of this case is two mobile users with a set of network operators in a global mobile telecommunications system. These two users, who are roaming between different networks, get security-related assistance from a set of network operators. Neither user trusts any individual network operator, but they may have reason to believe that some of the network operators are 'good'.

4. *Two clients only believe that a set of servers are not all colluding*. In this case, the clients do not trust any server, but they believe that all the servers cannot collude together to defraud them in some way. The example described in the last case still applies, where the mobile users may have reason to believe that the network operators are not all colluding.

The type of trust relationship which may exist between clients and servers depends on the environment in which the protocol is used. When there is no single server trusted by both clients, the clients possibly need to ask two or more untrustworthy servers to provide an authentication service. Such a set of servers, each of whom is of questionable trustworthiness when acting alone, can be believed by the clients to be acceptably trustworthy, when they are compelled to act jointly by an appropriate protocol. After the protocol succeeds, the trust in the joint action of all the servers is transferred to a trust in the security of the communications established using the session key between two clients.

Finally note that no protocol can work correctly if all the servers are untrustworthy and they all collaborate, since they can jointly impersonate one client to another. Hence the minimal possible trust requirement is encapsulated in Case 4 above.

In the next subsection, we will propose a new authentication and key distribution protocol using minimally trusted servers. This protocol works correctly unless all of the servers are untrustworthy and colluding.

4.5.3 The Protocol

4.5.3.1 Assumptions and notation

The protocol is based on symmetric encipherment and Diffie-Hellman key agreement. It is assumed that two clients A and B want to communicate securely with each other. For this purpose they need to verify the identity of one another and to establish a shared session key between them, but before the authentication processing starts they do not share any secret. They therefore make use of third parties, namely a set of servers S_1, ..., S_n. The trust relationship between A, B and S_1, ..., S_n can be any one of Cases 2, 3 and 4 defined in the previous section. In the protocol, A and B share secret keys K_{Ai} and K_{Bi} respectively with S_i.

We assume that the encipherment operation used provides origin authentication and integrity services, i.e. a received message encrypted using a shared secret must have been sent by the owner of the secret in the form that it was received.

We also assume that Diffie-Hellman key agreement used is based on a Galois field G with p elements (p prime), a set $N = \{1, ..., p\text{-}2\}$, and g, a primitive element of G. We suppose that A and B have agreed use of $g \in G$ which may be a universally used value within some domain of clients.

In the protocol descriptions, we make use of the following notation.

- $\{Z\}_K$ is the result of the encipherment of data Z with a symmetric encipherment algorithm using the key K.
- h is a one-way hash function.
- $D_Q \in N$ is entity Q's private key agreement value and R_Q (= $g^{D_Q} \bmod p$) is Q's public key agreement value. R_Q and D_Q are changed in every use of the protocol.
- $X \| Y$ is the result of the concatenation of data items X and Y.
- $P \rightarrow Q$: Z denotes that P sends message Z to Q.
- K_{AB} (=$h(g^{D_A D_B} \bmod p)$) is the session key shared by A and B as a result of the following protocol.

4.5.3.2 Exchanged messages

The protocol has the following exchanged messages, where messages **M2$_i$**, **M3$_i$**, **M5$_i$** and **M6$_i$** are sent for every i = 1, ..., n.

A initially chooses and stores D_A secretly, calculates $R_A = g^{D_A} \bmod p$, enciphers n blocks made up of R_A and B using K_{Ai} ($1 \le i \le n$), and sends these n blocks together with its own name to B in **M1**.

M1: $A \rightarrow B$: $A || \{R_A || B\}_{K_{A1}} || \{R_A || B\}_{K_{A2}} || \ldots || \{R_A || B\}_{K_{An}}$

Upon receiving **M1**, B chooses and stores D_B secretly, calculates $R_B = g^{D_B} \bmod p$, enciphers n blocks made up of R_B and A using K_{Bi} ($1 \le i \le n$), and sends the ith enciphered block with A's ith enciphered block and both clients' names to S_i in **M2$_i$** ($1 \le i \le n$).

M2$_i$: $B \rightarrow S_i$: $A || B || \{R_A || B\}_{K_{Ai}} || \{R_B || A\}_{K_{Bi}}$

On receipt of **M2$_i$**, S_i deciphers both parts, enciphers two blocks made up of R_A, R_B and A or B using K_{Bi} or K_{Ai} respectively, and then sends them in **M3$_i$** ($1 \le i \le n$) to B.

M3$_i$: $S_i \rightarrow B$: $\{R_A || R_B || B\}_{K_{Ai}} || \{R_B || R_A || A\}_{K_{Bi}}$

If an expected message from a particular server is not received by B during a time-out period, B eliminates this server and informs A. Note that to inform A which server has been eliminated, B can use an error notice in either **M4** or **M7** instead of the data block corresponding to that server. After receiving **M3$_i$**, B deciphers the blocks for him using K_{Bi}, and checks all the **M3$_i$** as follows.

B firstly checks whether all messages **M3$_i$** have a correct value of R_B. If not, B simply rejects any message with an incorrect value of R_B and informs A. B then checks whether the value of R_A in all the remaining messages **M3$_i$** are the same. If yes, B computes K_{AB} and $h(K_{AB} \| A)$, and sends **M4** to A.

M4: $B \rightarrow A$: $\{R_A || R_B || B\}_{K_{A1}} || \{R_A || R_B || B\}_{K_{A2}} || \ldots || \{R_A || R_B || B\}_{K_{An}} || h(K_{AB} || A)$

Otherwise, B will ask all the servers to provide a verification service that lets A and B decide which is a correct value of K_{AB}. Suppose B finds m different values of R_A in the messages **M3$_i$**, namely R_1, ..., R_m, which correspond to m different values of K_{AB}, namely K_1, ..., K_m ($K_j = h(R_j^{D_B} \bmod p)$). B assumes there should be one and only one correct value here. B sends S_i message **M5$_i$** ($1 \le i \le n$) to request such a service.

M5$_i$: $B \rightarrow S_i$: $A || B || m$

On receipt of **M5$_i$**, S_i ($1 \le i \le n$) chooses m random numbers T_{ij} ($1 \le j \le m$) and sends message **M6$_i$** ($1 \le i \le n$) to B.

M6$_i$: $S_i \rightarrow B$: $\{R_B || T_{i1} || T_{i2} || \ldots || T_{im} || A\}_{K_{Bi}} || \{R_A || T_{i1} || T_{i2} || \ldots || T_{im} || B\}_{K_{Ai}}$

After receiving $\mathbf{M6}_i$ ($1 \le i \le n$) or waiting for a time-out period (again, if an expected message from a particular server is not received by B during the time-out period, B eliminates this server and informs A in message **M7**), B sends message **M7** to A.

$$\begin{aligned}\mathbf{M7}:\ B \rightarrow A:\ & \{R_A||R_B||B\}_{K_{A1}}||\{R_A||R_B||B\}_{K_{A2}}||\ldots||\{R_A||R_B||B\}_{K_{An}}|| \\ & h(T_{11})||h(T_{12})||\ldots||h(T_{1m})||\{R_A||T_{11}||T_{12}||\ldots||T_{1m}\}_{K_{A1}}||\ldots \\ & h(T_{n1})||h(T_{n2})||\ldots||h(T_{nm})||\{R_A||T_{n1}||T_{n2}||\ldots||T_{nm}\}_{K_{An}}|| \\ & h(K_1) \oplus T_{11} \oplus T_{21} \oplus \ldots \oplus T_{n1}||\ldots \\ & h(K_m) \oplus T_{1m} \oplus T_{2m} \oplus \ldots \oplus T_{nm}\end{aligned}$$

Note that $h(T_{ij})$ ($1 \le i \le n$, $1 \le j \le m$) are hash values used to check if A and B receive the same random numbers.

After receiving either message **M4** or **M7**, A deciphers the first n blocks using K_{Ai} ($1 \le i \le n$) and checks them in a similar way as B did. A firstly rejects any message with an incorrect value of R_A. Then suppose A finds t ($1 \le t \le n$) different values of R_B in the remaining messages, which correspond to t different values of K_{AB}.

If A received **M4**, it computes all corresponding values of $h(K_{AB}||A)$, and compares them with the value B sent. If one value matches, A retrieves this value as the resulting session key K_{AB}, and then computes $h(K_{AB}||B)$ and sends message **M8** to B. Otherwise, A sends B an error message.

$$\mathbf{M8}:\ A \rightarrow B:\ h(K_{AB}||B)$$

On receipt of **M8**, B checks whether or not the same K_{AB} has been accepted by A.

If A received **M7** instead of **M4**, it checks whether all nonces of the servers match corresponding check values. If some values do not match, A informs B to eliminate the relevant servers and resend **M7**. Otherwise, A checks whether there is a value of K_j ($1 \le j \le m$) matching one of the computed values of K_{AB}. Note that in order to find such a correct value, A has to try all possibilities to compute

$$h(K_1) \oplus T_{11} \oplus T_{21} \oplus \ldots \oplus T_{n1},\ h(K_2) \oplus T_{12} \oplus T_{22} \oplus \ldots \oplus T_{n2},\ \ldots,\ h(K_m) \oplus T_{1m} \oplus T_{2m} \oplus \ldots \oplus T_{nm}$$

in **M7**. If one value does match, A sends B message **M9** and retrieves this value as a session key K_{AB}. Otherwise, A sends B an error message.

$$\mathbf{M9}:\ A \rightarrow B:\ h(K_{AB}||B)$$

After getting the 'positive' message in **M9**, B sends A the reply **M10** and retrieves the correct value of K_{AB}.

$$\mathbf{M10}:\ B \rightarrow A:\ h(K_{AB}||A)$$

4.5.4 Security analysis

Suppose the following four assumptions hold.

Assumption 4.4

A and B follow the protocol specifications correctly.

Assumption 4.5

The Diffie-Hellman key agreement parameters are chosen such that

a) computing D_A (D_B) from knowledge of g, p and R_A (R_B) is infeasible, and

b) computing K_{AB} from knowledge of g, p, R_A and R_B is infeasible.

Assumption 4.6

Communications channels used to transmit message exchanges during the protocol processing have the property that all entities involved in the protocol, including dishonest servers or other intruders, have the same access rights to the channels. That means a malicious entity can intercept messages, replay messages with or without modification, and even introduce new fraudulent messages to the channels, but cannot suppress messages. If any malicious entity interferes with communications between *A* and *B* or between *B* and the honest servers, *A*/*B* will request messages to be sent repeatedly until the protocol succeeds.

Assumption 4.7

In addition to Assumption 4.6, the 'bad' servers can only do the following:

a) fail to send messages to either *A* or *B*, or send messages with the wrong syntax;

b) send wrong public key agreement values R_A and/or R_B to *A* and/or *B*; and

c) send different nonces T_{ij} to *A* and *B*.

Then the protocol described in subsection 4.5.3 has the following properties.

Theorem 4.8

If at least one server follows the protocol, *A* and *B* can establish K_{AB}, which cannot be learnt by any server.

Proof

If at least one server follows the protocol specifications correctly, *B* gets at least one 'good' message including correct values of R_A and R_B in **M3**$_i$, and *A* also gets at least one 'good' message including the correct values of R_A and R_B in **M4** or **M7**. If there is no bogus value of R_A in **M3**$_i$, *A* and *B* can retrieve the correct values of R_A and R_B and agree on $K_{AB} = h(g^{D_A D_B} \bmod p)$ by following **M4** and **M8**.

If there are some bogus values of R_A in **M3**$_i$, *A* and *B* can follow **M5**$_i$, **M6**$_i$, **M7**, **M9** and **M10** to select a correct value of K_{AB}. The value $T_{1j} \oplus T_{2j} \oplus ... \oplus T_{nj}$ $(1 \le j \le m)$, which cannot be compromised by any server, is used to let *A* and *B* compare all the candidate values of K_{AB} and decide which value has been received by both of them and can be agreed on as the session key K_{AB}. Because of Assumption 4.5, all the servers can obtain R_A and R_B, but cannot discover K_{AB}.

Theorem 4.9

So long as not all servers are colluding, no matter whether there is a server following the protocol, no server can force *A*/*B* to accept a wrong message.

Proof

Suppose unless all servers follow the protocol correctly, or all servers are colluding, the probability of all the values (e.g. all bogus R_A, R_B and T_{ij}) agreeing is minimal. If all the servers follow the protocol incorrectly, but are not colluding, *B* might find more than one value of R_A in **M3**$_i$, the processing then moves to **M5**$_i$. Again, if not all the servers are colluding, no server can compromise $T_{1j} \oplus T_{2j} \oplus ... \oplus T_{nj}$ $(1 \le j \le m)$ and force *A* to accept a bogus message **M7** and then force *B* to accept a bogus message **M9**. The result is that although *A* and *B* might not agree on a correct value of K_{AB}, no server can force them to accept any wrong message.

Theorem 4.10

No server can learn K_{AB}, if the key is agreed by both *A* and *B*.

Proof

In any circumstance, including all the servers are 'bad' and colluding, if the same K_{AB} is obtained by both A and B, then it must be $h(g^{D_A D_B} \bmod p)$. However, there is no way that A (B) can verify that B (A) (rather than the colluding servers) possesses the value K_{AB}. During the protocol procedure, none of D_A, D_B and K_{AB} are communicated in any way. Because of Assumption 4.5, although any server can obtain R_A and R_B, it cannot discover D_A, D_B and K_{AB}.

Theorem 4.11

If any symmetric key K_{Ai} or K_{Bi} shared between A or B and S_i is compromised, no interceptor can discover the session key K_{AB} agreed by A and B.

Proof

During the protocol procedure, none of D_A, D_B and K_{AB} are communicated in any way. Because of Assumption 4.5, although any interceptor who has compromised K_{Ai}/K_{Bi} can obtain R_A and R_B, it cannot discover D_A, D_B and K_{AB}.

4.5.5 Relaxed trust requirements

Observe that the protocol described in subsection 4.5.3 works in Case 4 of trust relationships defined in subsection 4.5.2, i.e. A and B do not trust any server, and just believe that not all of the servers are colluding. The processing of the protocol can potentially be relaxed if A and B can trust the set of servers to a greater extent, e.g. in Case 2 or 3. To see this we give two examples.

Firstly, suppose A and B trust different but identified servers, e.g. their own service providers in a mobile telecommunications system, as defined in Case 2 of subsection 4.5.2. Note that A (and B) only knows which server is trusted by itself for some reason, but does not know which one is trusted by the other party. That is why they might still make use of n servers ($n > 2$), not two servers trusted by one of them respectively. A and B now only need to check whether or not a particular message from their own trustworthy server has been received; if it has, then any other message with a different R_B or R_A can be rejected and further processing will continue.

Secondly, suppose A and B believe that at least m servers are trustworthy (m is any integer from 1 to n), as defined in Case 3 of subsection 4.5.2. After receiving the messages $\mathbf{M3_i}$ (or waiting a time-out period if an expected message does not arrive), B checks and eliminates such messages which agree on a value of R_A and the number of which is smaller than m. The further processing then continues.

4.5.6 Stringent communications channels

We now suppose that, in some environment where the protocol described in subsection 4.5.3 is used, communications channels used to transmit message exchanges during the protocol processing might not meet Assumption 3 of subsection 4.5.4; in particular, dishonest servers might have greater rights than clients to access the channels.

Theorem 4.12

If dishonest servers can not only intercept exchanged messages but also suppress the messages, as well as insert, modify and replay the messages, they can subvert the protocol described in subsection 4.5.3 by impersonating one client to another.

Proof

Suppose there are n-1 honest servers and one dishonest server S_1. After receiving $\mathbf{M2_1}$, S_1 chooses and stores its private key agreement value D_X, and calculates the corresponding public key agreement value R_X ($= g^{D_X} \bmod p$). S_1 sends a bogus message $\mathbf{M3_1}$ to B.

$\mathbf{M3_1}$: $S_1 \rightarrow B$: $\{R_A || R_X || B\}_{K_{Ai}} || \{R_B || R_X || A\}_{K_{Bi}}$

In the meantime, S_1 simply suppresses all the messages $\mathbf{M3_i}$ between B and all the honest servers. As a result, B eliminates all the honest servers and informs A. On receipt of $\mathbf{M3_1}$, B

computes K_{BX} (= $h(R_X^{D_B} \bmod p)$) as a candidate value of K_{AB} and sends message **M4** with $h(K_{BX}||A)$ to A. S_1 also suppresses **M4** and modifies $h(K_{BX}||A)$ to $h(K_{AX}||A)$, where K_{AX} = $h(R_A^{D_X} \bmod p)$. On receipt of this bogus **M4**, A computes K_{AX} (= $h(R_X^{D_A} \bmod p)$) as a candidate value of K_{AB} and finds it matches the value $h(R_A^{D_X} \bmod p)$ from the bogus **M4**. A then sends $h(K_{AX}||B)$ in **M8** to B. Again, S_1 suppresses **M8** and modifies $h(K_{AX}||B)$ to $h(K_{BX}||B)$. B then retrieves K_{BX} as well. S_1 can now impersonate B to A by using K_{AX} with A and impersonate A to B by using K_{BX} with B.

In order to solve the stringent communications channel problem with the protocol described in subsection 4.5.3, we change that protocol by giving more stringent checks, again using exchanged messages **M1**, **M2**$_i$, **M3**$_i$, **M4** and **M8** of subsection 4.5.3. After receiving **M3**$_i$ ($1 \le i \le n$), B checks it in two ways:

1) any message with an incorrect value of R_B is rejected, and

2) the value of R_A in all n messages must be the same.

If any message is rejected by (1), or if (2) fails, then any further processing is rejected. On receipt of **M4**, A checks it in the same way as B did. Again, if any of the checks fail then all further processing is aborted. Note that this protocol will only succeed if all servers work correctly or all servers collude.

Theorem 4.13

The above protocol has the following properties.

a) If all the n servers follow the protocol, A and B can establish K_{AB}, which cannot be learnt by any server.

b) If at least one server follows the protocol, no server can force A/B to accept a wrong message.

c) If not all the n servers are colluding, no server can force A/B to accept a wrong message.

Proof

a) R_A and R_B can be correctly transferred to and accepted by B and A respectively, so that they can agree on K_{AB} = $h(g^{D_A D_B} \bmod p)$. Because of Assumption 4.5 in subsection 4.5.4, all the servers can obtain R_A and R_B, but cannot discover K_{AB}.

b) A and B get at least one 'good' message including correct values of R_A and R_B. If some server sends a wrong message, A/B will detect this since it will be inconsistent with the good message, and A/B will reject any further processing. Thus no server can force A/B to accept a wrong message.

c) Unless all the servers send messages with the same block made up of R_A, R_B and B to A, and the same block made up of R_B, R_A and A to B, A and B will reject further processing. Unless all servers follow the protocol correctly, or all servers are colluding, the probability of all the values agreeing is minimal.

4.5.7 K_{AB} 'strengthened' by servers

There may be a need for the session key K_{AB} to be 'strengthened' by servers. We give an example of a possible reason for this. In the previous protocol, if any symmetric key K_{Ai} shared between A and S_i is known to B, B could obtain R_A from **M1** and then control the resultant session key by selecting D_B. In order to prevent this, the servers may be asked to provide a contribution to the session key. In such a case, the following protocol is suggested, where messages **M2**$_i$, **M3**$_i$, **M5**$_i$ and **M6**$_i$ are repeated for i = 1, ..., n.

M1: $A \rightarrow B$: $A||\{R_A||B\}_{K_{A1}}||\{R_A||B\}_{K_{A2}}||\ldots||\{R_A||B\}_{K_{An}}$

M2$_i$: $B \rightarrow S_i: \; A||B||\{R_A||B\}_{K_{Ai}}||\{R_B||A\}_{K_{Bi}}$

M3$_i$: $S_i \rightarrow B: \; \{R_A||R_B||D_i||T_{i1}||T_{i2}||\ldots||T_{in}||B\}_{K_{Ai}}||\{R_B||R_A||D_i||T_{i1}||T_{i2}||\ldots||T_{in}||A\}_{K_{Bi}}$

M4: $B \rightarrow A: \; \{R_A||R_B||D_1||T_{11}||T_{12}||\ldots||T_{1n}||B\}_{K_{A1}}||\ldots$

$\{R_A||R_B||D_n||T_{n1}||T_{n2}||\ldots||T_{nn}||B\}_{K_{An}}||$

$h(K_1) \oplus T_{11} \oplus T_{21} \oplus \ldots \oplus T_{n1}||\ldots$

$h(K_n) \oplus T_{1n} \oplus T_{2n} \oplus \ldots \oplus T_{nn}$

M5: $A \rightarrow B: \; h(K_{AB}||B)$

After getting the 'positive' message in **M9**, *B* sends *A* the reply **M10** and retrieves the correct value of K_{AB}.

M6: $B \rightarrow A: \; h(K_{AB}||A)$

Each server generates a nonce D_i ($1 \le i \le n$) as a contribution to the session key. The candidate key K_i is defined by $K_i = g^{D_A D_B D_i} \mod p$. The hash values $h(K_1||B)$, ..., $h(K_n||B)$ and $h(K_1||A)$, ..., $h(K_n||A)$ are used to check whether or not *A* and *B* have retrieved the same candidate keys. If *B* (or *A*) think that some of the candidate keys K_i are unsuitable for use, he/she embeds an error message instead of $h(K_i||A)$ (or $h(K_i||B)$) to inform the other entity. The resultant session key is defined by $K_{AB} = h$(all agreed candidate keys) which, although contributed to and strengthened by the servers, is not disclosed to them.

4.5.8 Conclusions

We have analysed the possible trust relationships between clients and servers during authentication and key distribution, and proposed a protocol using minimally trusted servers. The protocol is based on symmetric cryptography and Diffie-Hellman key agreement. On the assumption that not all servers are untrustworthy and colluding, no untrustworthy server can subvert the protocol either by impersonating one client to another or compromising the security of communications between two clients. Even if the symmetric keys shared by clients and servers are compromised, any malicious third party (e.g. an interceptor) cannot discover the resultant session key, if it is agreed by two clients.

5. Service-related authentication

5.1 Introduction

In this section we consider a number of different aspects of the provision of service-related authentication for third generation systems.

In subsection 5.2 we consider two sets of general options for the provision of service-related authentication. The first set of options is defined within the context of the role model defined in TR1, [3], and the second set of options is defined in the context of the functional model, also given in TR1, [3]. These options distinguish between various parts which the role model or functional model entities might play within the authentication process. The subsection concludes by giving a matching between the two sets of options.

This leads naturally to the discussion in subsection 5.3 where protocols implementing service-related authentication for each of the role-model options are identified. The various cryptographic mechanisms which can be used to support these authentication protocols are discussed, and their relative merits evaluated.

Given that service-related authentication will typically need to support session key establishment, it is natural to next consider, in subsection 5.4, how the authentication protocols of subsection 5.3 can be adapted to provide such key distribution functions. It is important to distinguish the purpose of the distribution discussed here, from that considered in section 7 below. Here we are concerned primarily with key distribution across the air interface, i.e. to enable the provision of security services between the mobile user and the network. In section 7 the main focus is on end-to-end key distribution, i.e. to enable the provision of security services between a pair of mobile users.

5.2 Options for service-related authentication

5.2.1 Introduction

In this subsection we are concerned with the general properties of (service-related) authentication features in 3GS. That is, we consider general properties of the mechanisms underlying all authentication features directly related to the provision of service to one or more users. In the context of the security features listed in Section 2 of this report (and in Section 7.4.4 of Technical Report 1, [3]), this subsection is concerned with feature number 5.

We will concentrate on the authentication of the user to the 'network'. In this context, we will argue that, initially, the information required for authenticating a user is generated by a service provider. However, in certain circumstances authentication may be delegated to a network operator who then has to receive the appropriate information from the service provider.

The same arguments apply to authentication of the 'network' by the user. In our role model, the user has an agreement with a service provider, not with a network operator. Thus, the user has a basis for authenticating this service provider but may have to rely on the service provider to help authenticate a network operator, if that should be required.

We will not examine authentication between service providers and network operators as this does not seem to raise any issues specific to mobile systems. Moreover providing such authentication services appears to be relatively straightforward, and probably falls within the scope of present standardised protocols (notably ISO/IEC 9798, [4]).

The possible information flows for authentication mechanisms are considered with respect to two different models of 3GS, namely the role model and the functional model adopted by the 3GS3 project. In each case a number of general information flow options for the authentication process are identified and considered in some detail. The subsection concludes by comparing the sets of options derived from the two models.

Detailed authentication protocols implementing each of the identified options are described in subsection 5.3 of this report.

5.2.2 Authentication and the role model

5.2.2.1 Role model

There have been many role models proposed for 3GS. We use the role model of Section 3.2, Technical Report 1, [3]. The three roles from this model which are involved in service related authentication are Users, Network Operators (NOs) and Service Providers (SPs). They are defined in TR1 [3] as follows:

- **User**. A user is an entity that is authorised by a subscriber to use particular services subscribed to by the subscriber.
- **Network Operator**. A network operator is an entity that
    1. provides the network capabilities to support particular services, and
    2. allows users, using appropriate terminals, to gain access to the network in order to be able to use the services.
- **Service Provider**. A service provider is an entity that is responsible for the provision of particular services, and the associated database management.

For the purposes of this discussion of authentication we distinguish between a *new* network operator (responsible for providing the present visited network) and an *old* network operator (responsible for providing the previously visited network).

We also need to clarify relationships among users, network operators and service providers, with respect to the provision of authentication.

- **User/Network Operator relationship**. A user gets access to service(s) in 3GS by communicating with an NO. Before the NO provides a particular service to the user, the user's claimed identity will typically need to be authenticated. Moreover, before the user makes use of service(s) provided by the NO, the identity of the network operator will also typically need to be authenticated. Although the SP will need to be involved in some way during the authentication process(es), we assume throughout that during the authentication process the user directs all its communications to the NO.

- **User/Service Provider relationship**. A user is initially registered with one or more SPs. During this registration process the SP issues the user with a *User Identity* and other information to be used during the authentication process(es). In addition, the *user profile*, including service restrictions, security restrictions, and other user-related data, is registered by the SP.

  NOTE - In fact it could be argued that the NO and the user do not need to authenticate one another in the usual sense of the word. All the NO actually requires is assurance that, whoever the user is, the SP will be prepared to pay for the service that the NO provides to the user. Similarly, the user only needs assurance that the NO will provide the service requested at a cost agreeable to the user and with the quality of service (including security level, reliability, etc.) expected by the user.

- **Network Operator/Service Provider relationship**. When an NO communicates with a user, it will typically be necessary for the NO to obtain user-related information from the user's SP. The NO will also be responsible for providing user-related status information, including charge-related data, back to the SP.

5.2.2.2 Situations involving authentication

We next briefly discuss in what situations authentication services may be required.

The following mobility procedures could be considered as situations where authentication may be needed.

- **User initial location registration**. This procedure is initiated by the user to notify the current NO that the user is available for the receipt of calls (and other service provision). In this situation authentication both of the user to the NO and vice versa, is typically necessary. User initial location registration might result in the storage of user related data at the database of the current NO.

- **User final deregistration**. This procedure is initiated by the user to notify the present NO that the user will no longer be reachable. User deregistration might result in the deletion of user related data at the database of the current NO.

- **User location updating between different visited NOs**. This procedure happens when the user roams from the previous location area associated with the old NO to the present location area associated with the new NO. The procedure includes user registration to the new network operator and user deregistration from the old network operator. User location updating might result in the storage of user-related data at the database of the new NO and deletion of user-related data at the database of the old NO.

- **User detachment**. This procedure is initiated by the user to notify the present visited NO that the user is temporarily not reachable. It prevents unnecessary paging and it may activate appropriate forwarding services. As a result of this process the NO will modify the status information of the user in its database.

- **User attachment**. This procedure is initiated by the user to notify the present visited NO that the user is reachable again. As a result of this process the NO will modify the status information of the user in its database.

- **User outgoing call set-up**. This procedure is initiated by the user to notify the present visited NO that the user is going to make a call. Outgoing call set-up might result in both the NO and the user's SP modifying the status information of the user in their databases.

- **User incoming call set-up**. This procedure is initiated by the NO to inform the user that an incoming call is coming. Incoming call set-up might result in both the NO and the user's SP modifying the status information of the user in their databases.

Except for the situation of user location updating between different visited network operators, which involves four parties (user, SP, new NO and old NO), all the above procedures involve some or all of the three parties: user, SP and NO. It is important to note that in many of the situations mentioned above, the SP will need to be involved to assist the authentication process, even though the SP is not explicitly mentioned.

5.2.2.3 Role model related options for authentication

5.2.2.3.1 Initial assumptions

We make the following assumptions regarding the location of authentication information:

1. *At all times both a user and its SP have knowledge about the identity of each other, together with any associated authentication information.* We have already discussed this assumption in subsection 5.2.2.1.

2. *Before communication between a user and an NO starts, they have no knowledge about the identity of each other. After the user location registration with the NO, both user and NO retain appropriate authentication information throughout the visit of the user to the network.* More specifically (and in the context of the situations described in subsection 5.2.2.2) in the following situations where authentication can arise the user and NO have no prior authentication information for each other:

    - user initial location registration, and
    - user location updating.

    In the following cases the user and NO may have prior authentication information for each other:

    - user final deregistration,
    - user detachment,
    - user attachment,
    - user outgoing call set-up, and
    - user incoming call set-up.

3. *All relevant pairs of NOs and SPs have knowledge of the identity of one another, and all the information necessary to authenticate one another.* How this is to be achieved, and which other of the roles in the role model are involved, is not clear at present.

4. *All relevant pairs of NOs have knowledge of the identity of each other, and all the information necessary to authenticate one another.* How this is to be achieved, and which other of the roles in the role model are involved, is not clear at present.

5.2.2.3.2 Storage of authentication information

During authentication between a user and an NO, there are four different cases to consider, depending on where the authentication information about the user is held. These are as follows.

♦ Authentication information about the user is kept with the SP only, which may happen in the following situations:

- user initial location registration,
- user location updating when the old NO is not reachable or the authentication information hold in the old NO is not available, and

- other procedures defined in subsection 5.2.2.2, when the authentication information held in the present visited NO is not available.

♦ Authentication information about the user is kept with the old NO, which happens in the situation of location updating between different visited NOs when the authentication information held by the old NO is available.

♦ Authentication information about the user is kept with the current NO, which happens in the following situations:

- user final deregistration,
- user detachment,
- user attachment,
- user outgoing call set-up, and
- user incoming call set-up.

♦ Authentication information about the user is kept by the user, e.g. if asymmetric cryptographic techniques are used.

We now give four options for the authentication process, depending on who can provide authentication information about the user. It should be clear from the discussion above that these options are not meant to be mutually exclusive. It is very likely that different options will need to be used within the same system depending on the circumstances in which the authentication process is taking place.

- **R1** Authentication information about the user is provided by the SP.
- **R2** Authentication information about the user is provided by the old NO.
- **R3** Authentication information about the user is provided by the present visited NO.
- **R4** Authentication information about the user is provided by the user.

We now discuss these four options in more detail.

Before proceeding observe that we might also divide up various cases depending on where the user is going to obtain authentication information about the NO.

5.2.2.3.3 Authentication option R1

The first option is that the authentication information about the user is provided by the SP. During the authentication process, the NO communicates with the SP to get the necessary authentication information.

As has been discussed in subsection 5.2.2.1, because the user's identity is issued by the SP, the SP holds authentication information about the user permanently. So, theoretically, this option is always available. The advantage of using this option is that the SP keeps total control over the user's authentication information. The disadvantage of this option is the communications overhead between the NO and SP.

Given that Option R4 is not in use, this option has to be used in the following situations:

1. User initial location registration, and
2. any other mobility procedures defined in subsection 5.2.2.2 when the old NO or present visited NO cannot provide the necessary authentication information.

There are essentially two types of authentication protocol which meet the needs of this situation, depending on whether the SP or the NO actually performs the user identity verification.

- In the first type of protocol, the NO passes the user's authentication request to the SP. The SP verifies it and then sends the network operator a message to say that it was completed successfully or unsuccessfully. The NO essentially relays messages between the user and the SP.
- In the second type of protocol, the NO asks the SP for authentication information for the user, and then deals with the verification of user's identity by itself. An example of such a protocol is provided by the GSM scheme, where the 'visited network' (NO) holds challenge/response pairs provided by the 'home network' (SP). These pairs enable the NO to authenticate and re-authenticate a user without further reference to the home network (SP), until the pairs have been exhausted. However this latter scheme has the disadvantage that an NO could continue to re-use these pairs with the relevant user indefinitely, without reference to the home network (SP).

5.2.2.3.4 Authentication option R2

The second option is that the authentication information about the user is provided by the old NO. In this case, authentication information has to be transferred from the old NO to the new NO during the authentication process.

This option can only be selected in the situation of user location updating between different visited NOs, i.e. where the user roams from a location area under the control of one NO into a location area under the control of a new NO. The following conditions must be met.

1. The old and new NOs must be able to communicate securely with one another.
2. The authentication information held by the old NO is suitable for use by the new NO.

As in Option R1 above, there are two types of authentication protocol which may be used, depending on whether the new NO or the old NO actually verifies the identity of the user.

5.2.2.3.5 Authentication option R3

The third option is that authentication information about the user is already held by the current NO. This option could be used in the following situations:

1. user final deregistration,
2. user detachment,
3. user attachment,
4. user outgoing call set-up, and
5. user incoming call set-up.

A further possible case in which this option could be used is for re-authentication during the provision of a service, or prior to provision of an additional service.

Finally note that where a SP and an NO have close links, e.g. they are provided by the same organisation, the NO may have automatic access to authentication information for all the users subscribing to its associated SP. Clearly option R3 will apply in such circumstances.

5.2.2.3.6 Authentication option R4

The fourth option is that the authentication information about the user is provided by the user. In this case the authentication information is provided to the user by the SP at the time of subscription (and perhaps at intervals thereafter). During the authentication process, the new NO does not need to communicate with the SP or the old NO, since the necessary authentication information can be obtained directly from the user.

This option could be selected in any situation defined in section 5.2.2.2. However, it does require the use of strong authentication mechanisms, i.e. mechanisms where the information presented by the claimant cannot be misused by the verifier, e.g. for impersonating the client. Certificates containing the public key of the claimant signed by a *Certification Authority* are already employed in some distributed systems such as Digital's DSSA, see, for example, Tardo and Alagappan's paper, [39].

In our case, the SP seems to be a natural candidate for the certification authority. It could issue authentication tokens to its users and give the key for verifying these tokens to the NOs it deals with.

Presumably, similar mechanisms could be used to provide authentication of the NO to the user, although these questions will only be completely resolved when detailed protocols are devised.

The disadvantage of this approach is that the cryptographic techniques required tend to be computationally intensive, and require user hardware to be relatively complex (and expensive). Another disadvantage may be the fact that the SP loses some control over the actions of its subscribers. More complicated revocation mechanisms may be required than in a situation where the SP can authorise each individual service request. The major advantage of this approach is the potential reduction in communication overheads.

5.2.3 Authentication and the functional model

We now consider how authentication might operate with respect to the 3GS functional model.

5.2.3.1 Functional model

Section 4.2 of Technical Report 1, [3], defines the generic IN functional model, currently agreed for use within the 3GS3 project. This model outlines the types of functional entities (FEs) in a mobile station and network, and shows the functional relationships between these FEs. The FEs are grouped into three classes:

- **Service Management**, which includes functions related to service creation, service provision, customer control capabilities, and support for the administration, co-ordination and control of a database,
- **Intelligence**, which includes functions related to service logic and service control, and
- **Access and Transport**, which includes functions related to access, call and bearer control.

For the purposes of this subsection, we make the assumption that all activities relating to authentication take place at the intelligence class of the functional model, with the exception that certain authentication related data and management signals are transferred via FEs of other classes. Following the model, we also restrict our attention to the Access Network (the serving network for a mobile originated call) and the Service Provider of the mobile subscriber.

Because of this assumption we restrict our discussion to FEs of the Intelligence Class. Of these FEs we restrict our attention to six types, namely two in the mobile part:

- the Mobile Storage Function, **MSF**, and
- the Mobile Control Function, **MCF**,

and four within the network part:

- the Service Data Function (Mobile) in the Network Operator, or **SDF(M)n**,
- the Service Control Function (Mobile) in the Network Operator, or **SCF(M)n**,
- the Service Data Function (Mobile) in the Service Provider, or **SDF(M)s**, and
- the Service Control Function (Mobile) in the Service Provider, or **SCF(M)s**.

The main authentication-related functions of these six types of FE are as follows.

♦ **MSF**. An **MSF** is a pure data storage function at the mobile side of the radio interface. It stores subscription or service related parameters, location information, and identity and security related parameters. In connection with service-related authentication, an **MSF** stores the following data items on a permanent basis (see also Appendix A of Technical Report 1, [3]):

- the IMUI (International Mobile User Identity),

- one or more authentication-related keys,
- a list of allowed services,
- security restrictions,
- default security, and
- a priority list of access domains.

In addition, an **MSF** stores the following data items on a temporary basis:

- a TMUI (Temporary Mobile User Identity),
- session keys and/or other related keys, and
- location registration information.

♦ **MCF**. An **MCF** contains the service logic and service-related processing required at the mobile side of the radio interface. In general it includes the following functionalities:

- network information monitoring and analysis,
- location update initiation,
- authentication processing,
- confidentiality control, and
- paging recognition and response.

In connection with service related authentication, an **MCF** performs the following processes:

- it accesses data in the **MSF**;
- it performs authentication related algorithms, and
- it shares control with the **SCF(M)n**.

♦ **SDF(M)n** and **SDF(M)s**. An **SDF(M)** handles storage and access to service related data and network data, and provides consistency checks on data. It hides the real data implementation from the **SCF(M)** and provides a logical data view to the **SCF(M)**. In general it includes functionalities to store service and mobility related data (e.g. location information, service profile and security related parameters), check data consistency, and initiate data updating.

In connection with service related authentication, an **SDF(M)s** stores the following data items on a permanent basis:

- service provider identity;
- authentication related keys;
- map of user numbers to identities;
- user service restrictions;
- user security restrictions; and
- related agreements.

In addition, the **SDF(M)s** stores the following data items on a temporary basis:

- mapping between user numbers (IMUIs) and TMUIs,
- mapping between user numbers (IMUIs) and location information, and
- session keys and/or related authentication messages.

In connection with service related authentication, an **SDF(M)n** stores the following data items on a permanent basis:

- network operator identity, and
- authentication related keys.

In addition, the **SDF(M)n** stores the following data items on a temporary basis:

- session keys and/or related authentication messages,
- list of services to be activated, and
- location registration information.

♦ **SCF(M)n** and **SCF(M)s**. An **SCF(M)** contains the overall service and mobility control logic and handles service related processing activity. It generally includes the following functionalities:

- paging control,
- service feature analysis,
- routing information provision,
- location management,
- identity management,
- user verification and authentication,
- authentication processing, and
- confidentiality control.

In connection with service-related authentication, an **SCF(M)n** and an **SCF(M)s** perform the following processes:

- access to data in **SDF(M)n** and **SDF(M)s**,
- share control, and
- perform authentication-related algorithms.

5.2.3.2 Functional model related options for authentication

Within the intelligence class of the functional model, the control of inter-working between a mobile station and the fixed network is shared between the **MCF** (within the mobile side) and the **SCF(M)n** (within the network side). The authentication process may involve the use of data stored in any of three locations: the **MSF**, the **SDF(M)n** and the **SDF(M)s**. Hence we first consider possible means by which data may be transferred between networks.

5.2.3.2.1 Options for data access between networks

Suppose there are two networks, each with its own **SDF(M)** and **SCF(M)** Functional Entity. Label the FEs in the first network as $\mathbf{SDF(M)_1}$ and $\mathbf{SCF(M)_1}$, and those in the second network as $\mathbf{SDF(M)_2}$ and $\mathbf{SCF(M)_2}$. Now, there are a number of different methods for controlling inter-working between two networks (e.g. the Network Operator and the Service Provider or a local and a remote Network Operator). Options for such control include the following.

1. The **SCF(M)** in one network can directly access the **SDF(M)** in a different network. If this is to take place then access control procedures will need to be strictly enforced. In this case the access paths between FEs for data access would be of the following type:

   $\mathbf{SCF(M)_1}$ - $\mathbf{SDF(M)_2}$ or $\mathbf{SCF(M)_2}$ - $\mathbf{SDF(M)_1}$

2. The **SCF(M)** in each of the two networks can share control and provide controlled access to each other's **SDF(M)** functional entities. In this case the access paths between FEs for data access would be of the following type:

   $SCF(M)_1$ - $SCF(M)_2$ - $SDF(M)_2$ or **$SCF(M)_2$ - $SCF(M)_1$ - $SDF(M)_1$**

3. The **SDF(M)** FEs in the two networks can co-operate to function as a distributed database, and hence an **SCF(M)** will request any data from its local **SDF(M)**, and the **SDF(M)** will be responsible for retrieving data as necessary from other **SDF(M)** FEs. In this case the access paths between FEs for data access would be of the following type:

$SCF(M)_1$ - $SDF(M)_1$ - $SDF(M)_2$ or **$SCF(M)_2$ - $SDF(M)_2$ - $SDF(M)_1$**

5.2.3.2.2 Four functional model options for authentication

According to which Functional Entities are involved, we can identify the following four functional model related options for authentication.

- **F1** Authentication involving **MSF**, **MCF**, **SDF(M)n** and **SCF(M)n**.
- **F2** Authentication involving **MSF**, **MCF**, **SDF(M)s** and **SCF(M)s**.
- **F3** Authentication involving **MSF**, **MCF**, **SDF(M)n**, **SCF(M)n**, **SDF(M)s** and **SCF(M)s**.
- **F4** Authentication involving **MSF**, **MCF**, **SDF(M)n**, **SCF(M)n**, **SDF(M)n'** and **SCF(M)n'**, where the **n'** FEs belong to a previously visited Access Network.

We now consider each of these four options in more detail.

5.2.3.2.3 Authentication option F1

In this option, authentication involves the four FEs: **MSF**, **MCF**, **SDF(M)n** and **SCF(M)n**. Authentication related data are stored in the **MSF** and the **SDF(M)n** throughout the duration of the mobile user's registration in the Network Operator. Authentication processing is performed by shared control between the **MCF** and the **SCF(M)n**.

This option requires one of the following two statements to hold:

- information necessary to authenticate the mobile user must be held in the **SDF(M)n** before the authentication process starts, or
- the **MSF** is capable of providing all the necessary authentication information to the **SCF(M)n**.

This option could be selected in the following cases:

1. user final deregistration,
2. user detachment,
3. user attachment,
4. user outgoing call set-up,
5. user incoming call set-up, and
6. using a strong authentication method, e.g. based on a digital signature algorithm or a zero-knowledge protocol.

5.2.3.2.4 Authentication option F2

In this option, authentication involves the four FEs: **MSF**, **MCF**, **SDF(M)s** and **SCF(M)s**. Authentication-related data are stored in the **MSF** and the **SDF(M)s**. Authentication processing is performed by shared control between the **MCF** and the **SCF(M)s**.

The **SDF(M)n** has no role here, and the **SCF(M)n** just plays a transmitting role between the **MCF** and the **SCF(M)s**. It is possible that another Functional Entity in the Network Operator, such as the Service Switching Function (SSF), plays this transmitting role instead of **SCF(M)n**.

This option could be selected in any situation defined in subsection 5.2.2.2.

5.2.3.2.5 Authentication option F3

In this option, authentication involves the six FEs: **MSF**, **MCF**, **SDF(M)n**, **SCF(M)n**, **SDF(M)s** and **SCF(M)s**. Authentication-related data are stored in the **MSF** and the **SDF(M)s**. Authentication processing is performed by shared control between the **MCF** and the **SCF(M)n**. The Service Provider Functional Entities (**SDF(M)s** and/or **SCF(M)s**) are responsible for providing authentication-related data to the **SCF(M)n**.

There are three different ways in which data can be delivered from the **SDF(M)s** to the **SCF(M)n**, corresponding to the three data access options discussed in subsection 5.2.3.2.1 above.

1. Direct access to stored Service Provider data:

   SCF(M)n - SDF(M)s.

2. Shared control between **SCF(M)n** and **SCF(M)s**:

   SCF(M)n - SCF(M)s - SDF(M)s

3. Distributed databases:

SCF(M)n - SDF(M)n - SDF(M)s

This authentication option could be selected in any situation defined in subsection 5.2.2.2.

5.2.3.2.6 Authentication option F4

In this option, authentication involves the six FEs: **MSF**, **MCF**, **SDF(M)n**, **SCF(M)n**, **SDF(M)n'** and **SCF(M)n'**, where the **n'** FEs belong to a Network Operator previously visited by the mobile user. Authentication-related data are stored in the **MSF** and the **SDF(M)n'**. Authentication processing is performed by shared control between the **MCF** and the **SCF(M)n**. The **SDF(M)n'** and/or **SCF(M)n'** are responsible for providing authentication-related data to the **SCF(M)n**.

Exactly analogously to the previous option, there are three different ways in which data can be delivered from the **SDF(M)n'** to the **SCF(M)n**, corresponding to the three data access options discussed in subsection 5.2.3.2.1.

1. Direct access to stored Network Operator data:

   SCF(M)n - SDF(M)n'.

2. Shared control between **SCF(M)n** and **SCF(M)s**:

   SCF(M)n - SCF(M)n' - SDF(M)n'

3. Distributed databases:

SCF(M)n - SDF(M)n - SDF(M)n'

This option could be selected for user location updating between different visited network operators defined in subsection 5.2.2.2.

5.2.4 Relationship between the two sets of options

We now briefly consider the relationships between the two sets of authentication options introduced in subsections 5.2.2.3 and 5.2.3.2.

We first note the following relationships between *roles* and *functions*.

- A user corresponds to the **MSF** and the **MCF** within a mobile station.
- A network operator corresponds to **SDF(M)n** and **SCF(M)n**.
- A service provider corresponds to **SDF(M)s** and **SCF(M)s**.

This then leads us to observe the following relationships between the two kinds of authentication options.

- Option R1 corresponds to Options F2 and F3.
- Option R2 corresponds to Option F4.
- Options R3 and R4 correspond to Option F1.

5.2.5 Trust and authentication

There are subtle trust implications implicit in each of the authentication options which have been discussed in the text above. For example, in any role model option the user will clearly need to trust their SP, with which the user will have a contractual arrangement. However, the user may also need to trust one or more NOs (for example in Role Model option R2, where the SP is not directly involved in the authentication process). The user will not have a contractual relationship with these NOs, and hence may be less willing to trust the NOs completely.

Ideally the selected authentication protocols should minimise the amount of trust that the user needs to place in NOs. This issue has been considered in detail in subsections 4.4 and 4.5.

5.3 Protocols for service-related authentication

5.3.1 Introduction

This subsection discusses authentication protocols for implementing role model related authentication options in third generation mobile telecommunications systems (3GS), as identified in subsection 5.2. Each of the authentication options could be implemented using a number of different protocols, and we discuss some possibilities below. Most of the protocols in this section can be found in the existing ISO authentication protocol standard (ISO/IEC 9798, [4]) or published papers (as referenced in the text below). One is selected from the second generation mobile telecommunications system GSM.

A definition of service related authentication in 3GS is given in the previous subsection. The three roles which are involved in service related authentication, as identified there, are Users, Network Operators (NOs) and Service Providers (SPs). The following four options for the authentication process have been described in subsection 5.2, and are based upon which role provides authentication information about the user.

- **R1** Authentication information about the user is provided by the SP;
- **R2** Authentication information about the user is provided by the old NO;
- **R3** Authentication information about the user is provided by the present visited NO; and
- **R4** Authentication information about the user is provided by the user.

For the purposes of this subsection, entity authentication protocols are divided into the following four categories:

- **P1** Entity authentication using symmetric encryption techniques;
- **P2** Entity authentication using a public key algorithm;
- **P3** Entity authentication using a cryptographic check function; and
- **P4** Entity authentication using zero-knowledge techniques.

Examples of protocols in categories P1, P2 and P3 have been defined in ISO/IEC 9798 parts 2, 3, and 4, [4]. Protocol category P4 will be covered by ISO/IEC 9798-5, [4], which is currently at an early stage of development; this section does not cover P4, although it appears to be a topic worthy of future study (see also section 11 below).

All the authentication protocols considered here use *Time Variant Parameters (TVPs)* to provide freshness/timeliness verification for messages making up the protocols. There are three main types of TVP, namely time stamps, sequence numbers and random numbers. For the purpose of this document we consider only nonce-based protocols, following the examples of GSM and DECT. If third generation systems should require mechanisms based on time stamps or sequence numbers, then appropriate protocols can be found relatively easily.

Service related authentication is a prelude to further communication between a user and an NO, so an authentication protocol at least must complete identity verification, i.e., the mutual verification of the identities of the user and NO. In addition, the protocol can optionally achieve key distribution and key confirmation. Key distribution means the user and NO agree upon a session key for later use (e.g. within a user session or a service session). Key confirmation means the user and NO both believe that each other has received the session key properly and believes in the suitability of the key. We concentrate here on a discussion of identity verification. Although key distribution and key conformation are not the main focus of this subsection, it is possible for a number of the protocols described here to achieve them by using the text fields included in the exchange messages appropriately. In other words, if a session key needs to be generated during the authentication process, it can be distributed as a part of the text fields.

5.3.2 A comparison of different cryptographic techniques

Before we discuss which category of protocols could be used for the implementation of which authentication option, we give a general analysis of the advantages and disadvantages of each category of protocols.

The main advantages of P1 algorithms relative to P2 algorithms are as follows.

1. Symmetric algorithms are computationally much less expensive than asymmetric ones, given the current stage of development of asymmetric algorithms.
2. Symmetric algorithm responses are short (typically 32 - 64 bits) in comparison with the asymmetric algorithm signatures (512 - 1000 bits), see also subsection 4.2.2.4.
3. It is perhaps simpler for symmetric encryption to distribute session keys during the authentication process as compared with P2.

The main advantages of P1 algorithms relative to P3 algorithms are as follows.

1. It is perhaps simpler for symmetric encryption to distribute session keys during the authentication process as compared with P3.

The main advantages of P3 algorithms relative to P1 algorithms are as follows.

1. It is a well-established part of cryptographic folk-lore that, although all cryptographic algorithms fall under the COCOM regulations and hence implementations of such algorithms require export licences from all the COCOM countries, systems incorporating cryptographic check functions are much more readily exported than systems using encryption. This means that P3 systems are potentially much more attractive to systems implementors.
2. It is possible that P3 algorithms are computationally simpler than P1 algorithms, although this distinction is not as clear as that between symmetric and asymmetric algorithms.

The main advantages of P3 algorithms relative to P2 algorithms are as follows.

1. Symmetric algorithms are computationally much less expensive than asymmetric ones, given the current stage of development of asymmetric algorithms.

The main disadvantage of P1 and P3 is that it seems very hard to let all relevant pairs of NOs, SPs and NOs share a secret key in 3GS, in particular, when a new subsystem, e.g. a new country or area network is affiliated with 3GS, the distribution and management of new shared keys cost a lot.

This disadvantage of P1 and P3 indicates the advantage of P2, i.e. in comparison with P1 and P3, it is easy to distribute public keys of every SP and NO to another.

The main disadvantages of P2 are as follows:

1. It is typically much more computationally expensive than P1 and P3.
2. It needs much longer asymmetric algorithm signature data to be transferred on the radio path than P1.
3. It is more difficult for P2 to distribute session keys during the authentication process than P1.
4. Asymmetric algorithms are rare.

The possibility and methods of key distribution in P2 and P3 are beyond the scope of this report, although it is an interesting topic worthy to be considered at another report in the near future.

One or two instances of the protocols for the implementation of each of the options will be described in the subsections below, including a brief discussion of the advantages and disadvantages of every instance.

5.3.3 A protocol for authentication option R1

In this option, the authentication information about the user is provided by the SP, who has knowledge of the identities of both the user and the NO. Before communication between the user and NO starts, they have no knowledge about the identity of each other, but have knowledge about the SP's identity. During the authentication process, two communications channels are available, namely those between the NO and SP, and the user and the NO. The NO communicates with the SP to get the necessary authentication information.

In theory, protocols from each of categories P1, P2 and P3 could be used in this situation. However, as has been discussed in subsection 5.3.1, the obvious advantages of P1 are simpler algorithms, shorter transferred data and easier methods of session key distribution; the main advantage of P3 is that algorithms of P3 are simpler than P2 and might even be simpler than P1. Probably the most serious flaw of P1 and P3 is to have to require all relevant pairs of SPs and NOs to share a secret key which will result in increased complexity for key distribution and management in 3GS. The main advantage of P2 is that is not necessary to distribute a shared key between every relevant pair of NO and SP, but the main disadvantage of P2 is that the asymmetric algorithms required tend to be much more computationally expensive. Hence the choice of which protocol category should be used for R1 must be made very carefully. As an example we present a protocol based on symmetric encryption (i.e. option P1). In practice, options P2 and P3 may be preferable.

We now describe a modified version of a shared-key nonce-based protocol taken from ISO/IEC 9798-2, [4]. In the language of that standard it is a five pass mutual authentication mechanism involving a trusted third party.

As we have already mentioned, the following protocol is of type P1 for authentication option R1 (see subsection 5.2.2.3.3). In connection with this protocol, all entities involved (the user, NO and SP) know the distinguishing names *U*, *N*, *S* of the user, NO and SP respectively. The SP and user share a secret key K_{SU}, and the SP and NO share another secret key K_{SN}.

M1: $U \rightarrow N$: $U||N||R_U||D_1$

M2: $N \rightarrow S$: $U||N||R_U||R_N||D_2$

M3: $S \rightarrow N$: $D_5||e_{K_{SN}}(U||R_N||K_{UN}||D_4)||e_{K_{SU}}(N||R_U||K_{UN}||D_3)$

M4: $N \rightarrow U$: $D_7||e_{K_{SU}}(N||R_U||K_{UN}||D_3)||e_{K_{UN}}(U||R_U||R_N'||D_6)$

M5: $U \rightarrow N$: $D_9||e_{K_{UN}}(N||R_N'||R_U||D_8)$

where R_U, R_N and R'_N are three nonces generated by user *U* and NO *N* respectively; D_i is the *i*th optional text field; K_{UN} is a session key for user *U* and NO *N*; $e_K()$ denotes encryption using the key *K*; and *A*||*B*||*C*||... represents the concatenation of data strings *A*, *B*, *C*, etc. **in the specified order**. The terminology defined here will be used throughout this subsection.

The differences between this protocol and the protocol in ISO/IEC 9798-2, [4], are in two minor changes made to the fourth and fifth messages **M4** and **M5**, namely changing $e_{K_{UN}}(R_N'||R_U||D_6)$ into $e_{K_{UN}}(U||R_U||R_N'||D_6)$ and $e_{K_{UN}}(R_U||R_N'||D_8)$ into $e_{K_{UN}}(N||R_N'||R_U||D_8)$. A change of this type (albeit with some re-ordering of the data fields) can be made conformant to the ISO/IEC specification, by appropriate use of the data fields D_6 and D_8.

The procedural description of the protocol is as follows.

1. User *U* sends NO *N* a message **M1** containing the distinguishing names *U* and *N*, a nonce R_U, and, optionally, a text field *D1*.

2. On receipt of **M1**, NO *N* determines the appropriate SP *S* from some combination of *U* and the data string D_1, and then sends the SP *S* a message **M2** including the distinguishing names *U* and *N*, the nonces R_U and R_N, and, optionally, a text field D_2.

3. After receiving **M2**, the SP *S* chooses a secret authentication key K_{UN} for user *U* and NO *N*, then returns the NO *N* a message **M3** including a ticket for *U*, namely $e_{K_{SU}}(N||R_U||K_{UN}||D_3)$.

4. On receipt of **M3**, the NO *N* decrypts the first part using the key K_{SN}, and from checking the correctness of *U* and R_N, *N* verifies that message **M3** was from *S* and was a reply to **M2**. NO *N* then retrieves K_{UN}, and forwards the ticket concatenated with another message encrypted using K_{UN} to *U*.

5. On receipt of **M4**, user *U* decrypts the ticket using the key K_{SU}, and by checking the correctness of *N* and R_U, user *U* verifies the ticket was issued by *S* and was a reply to **M1**. User *U* retrieves the key K_{UN}, and then, after using K_{UN} to decrypt the other part of **M4** and checking the correctness of *U* and R_U, user *U* believes NO *N* has retrieved K_{UN}. User *U* then returns **M5** to NO *N*.

6. On receipt of **M5**, NO *N* decrypts **M5** using the key K_{UN}. By checking the correctness of *N*, R_U and R'_N, NO *N* verifies that **M5** was sent by user *U* and was a reply to **M4**. NO *N* can also be sure that *U* has retrieved the key K_{UN}.

This protocol enables the user and the NO to verify each other's identity, and, in addition, the user and the NO agree upon the secret authentication key K_{UN}, and the user and NO know that the key K_{UN} is possessed by each other. Optionally the key K_{UN} can be used as a session key for the provision of additional security features (e.g. data confidentiality) in subsequent communications between the user and the NO (see also subsection 5.4).

Besides those discussed in subsection 5.3.1, another advantage of this protocol is:

- The user specific key K_{SU} is known only to the user and SP; it does not have to be revealed to the NO. Similarly, the NO specific key K_{SN} is known only to the NO and SP; it does not have to be revealed to the user.

Other disadvantages of this protocol are:

- The NO has to communicate with the SP in order to obtain the authentication information.
- The authentication algorithm has to be known to the NO, so it needs to be standardised.
- The NO must preserve the confidentiality of the authentication key K_{UN}.

5.3.4 A protocol for authentication option R2

In authentication option R2, the authentication information about the user is provided by the old NO. In this case, authentication information has to be transferred from the old NO to the new NO during the authentication process. NO_o and NO_n are identified as an old network operator (responsible for providing the previously visited network) and a new network operator (responsible for providing the present visited network) respectively.

Given that in this role model option we have two entities authenticating one another with the help of a third party, R2 is quite similar to R1. In fact, if we change SP into NO_o and NO into NO_n in the protocol described in subsection 5.3.3, we can get an example of a protocols for R2. As for role model option R1, protocols in categories P1, P2 and P3 could be used. In addition to the advantages and disadvantages described in subsection 5.3.3 above, the trouble with using protocols of types P1 and P3 is that all relevant pairs of NOs must share a secret key. Whether or not P2 is a viable candidate depends on whether secret key distribution and management between NOs is too difficult or costs too much.

The main difference from R1 is that the 'authentication server' NO_o in R2 is not the party who originally issued the user's authentication information. The information about the user held by NO_o was issued by the SP at some previous time (perhaps using an authentication mechanism falling under the heading of role model R1). In this subsection we describe one example of a protocol using a

cryptographic check function (i.e. of type P3); this protocol is currently used in GSM, [40]. If we change NO_o into SP and NO_n into NO in this protocol, we obtain a protocol of type P3 implementing role model option R1.

The following protocol is a unilateral authentication protocol, in which only NO_n can verify the user's identity by using the authentication information transferred from NO_o. The user is not provided with the means to verify the identity of NO_n. In the protocol description below, *N* and *N'* represent the new NO and old NO's distinguishing names respectively.

M1: $U \rightarrow N: \ D_1 \| U \| N \| N'$

M2: $N \rightarrow N': \ D_2 \| U \| N \| N'$

M3: $N' \rightarrow N: \ D_3 \| U \| N \| N' \| RAND \| c_{K_i}(RAND)$

M4: $N \rightarrow U: \ D_4 \| RAND$

M5: $U \rightarrow N: \ D_5 \| c_{K_i}(RAND)$

where c_K denotes a cryptographic check function using the key *K*.

In fact, in GSM only the user and corresponding SP hold the user-specific key K_i and the cryptographic check function c_{K_i}; the NO does not require knowledge of either of them. All challenges *RAND* and responses $c_{K_i}(RAND)$ (labelled *SRES* in GSM) are issued by the SP when an NO asks the SP for authentication information about the user at the former authentication process. So the NO_o (*N'*) and NO_n (*N*) here just forward the authentication messages between the SP and user.

The procedural description of this protocol is as follows:

1. *U* sends the distinguishing names *U*, *N* and *N'*, and, optionally, a text field D_1 to *N*.
2. After receiving **M1**, *N* sends the same distinguishing names, and, optionally, a text field D_2 to *N'*.
3. On receipt of **M2**, *N'* sends *N* **M3** containing a challenge *RAND* and a response to the challenge $c_{K_i}(RAND)$ (i.e. what in GSM is called *SRES*).
4. On receipt of **M3**, *N* forwards the challenge to *U* in message **M4**.
5. On receipt of *RAND* (in **M4**), *U* generates *SRES* using the key K_i, and then sends **M5** containing *SRES* to *N*.
6. On receipt of **M5**, *N* compares the two *SRES*s sent by *U* and *N'*, thereby verifying that **M5** was a reply to **M4** and was issued by *U*.

Besides those discussed in subsection 5.3.1, other advantages of this protocol are:

1. The user specific key K_i is known only to the user and SP, and it does not have to be revealed to the NO.
2. The authentication algorithm does not have to be known to the NO, so it does not need to be standardised.

Other disadvantages of this protocol are:

1. NO_n has to communicate with NO_o in order to obtain the authentication information.
2. The NO must preserve the confidentiality of the authentication information.
3. The identity of the NO is not verified to the user or the user's SP.

5.3.5 A protocol for authentication option R3

In this option, the authentication information about the user is already held by the current NO. As previously, protocols in any of categories P1, P2 and P3 could be used for the implementation of this authentication option. In comparison with other categories, P1 and P3 require simpler algorithms but require the maintenance of shared secret keys; for P2 is not necessary to maintain shared secret keys but the algorithms tend to be much more computationally expensive. The choice of category remains a difficult question. We now give one example of a protocol using symmetric encryption (i.e. a protocol of type P1).

The following type P1 protocol implementing R3 is the three pass mutual authentication protocol from ISO/IEC 9798-2, [4]. In this protocol it is assumed that the user and NO both know their respective distinguishing names *U* and *N*, and also a shared secret key K_{UN}.

M1: $U \rightarrow N: \; D_1 \| U \| N \| R_U$

M2: $N \rightarrow U: \; D_3 \| e_{K_{UN}}(U \| R_U \| R_N \| D_2)$

M3: $U \rightarrow N: \; D_5 \| e_{K_{UN}}(N \| R_N \| R_U \| D_4)$

The procedural description of this protocol is as follows:

1. *U* sends the distinguishing names *U* and *N*, a nonce R_U, and ,optionally, a text field D_1 to *N*.
2. After receiving **M1**, *N* sends **M2** to *U*.
3. On receipt of **M2**, *U* decrypts **M2** using the key K_{UN}, and, by checking the correctness of *U* and R_U, *U* is able to verify that **M2** was sent by *N* and was a reply to **M1**. Finally *U* sends **M3** to *N*.
4. On receipt of **M3**, *N* decrypts **M3** using K_{UN}, and, by checking the correctness of *N*, R_U and R_N, *N* verifies that **M3** was sent by *U* and was a reply to **M2**.

This protocol enables the user and the NO to verify each other's identity. If a session key needs to be generated during the authentication process, the key can be distributed as a part of the text fields.

In addition to those discussed in subsection 5.3.1, one other advantage of this protocol is:

1. The NO does not have to communicate with the SP in order to obtain the authentication information.

Other disadvantages of this protocol are:

1. The authentication algorithm has to be known to the NO, so it needs to be standardised.
2. The NO must preserve the confidentiality of the authentication key K_{UN}.

5.3.6 A protocol for authentication option R4

In this option, the authentication information about the user is provided by the user. Because the user and NO have no shared secret key, only protocols of type P2 (or perhaps P4) can be used here. This subsection describes a protocol using public key techniques (i.e. of type P2) implementing R3. This protocol is the three-pass mutual authentication mechanism from ISO/IEC 9798-3, [4].

M1: $U \rightarrow N: \; D_1 \| U \| N \| R_U$

M2: $N \rightarrow U: \; D_3 \| U \| R_U \| R_N \| s_{S_N}(U \| R_U \| R_N \| D_2)$

M3: $U \rightarrow N: \; D_5 \| N \| R_N \| R_U \| s_{S_U}(N \| R_N \| R_U \| D_4)$

where $s_K()$ denotes a signature computed using the key *K*, S_N denotes the secret signature key of *N*, and S_U denotes the secret signature key of *U*. Optionally, data strings D_3 and D_5 may contain

certificates for the public keys of *N* and *U* respectively. The recipient of such a certificate can use it to obtain a validated copy of the public key of the owner of the certificate.

The procedural description of this protocol is as follows:

1. *U* sends the distinguishing names *U* and *N*, a nonce R_U, and, optionally, a text field D_1 to *N*.
2. After receiving **M1**, *N* send *U* the message **M2** (optionally including a certificate for the public signature verification key of *N*).
3. On receipt of **M2**, *U* verifies the signature of *N* using *N*'s public verification key, and checks the correctness of *U*, R_U and R_N, thereby verifying that **M2** was sent by *N* and was a reply to **M1**. *U* may obtain the public key of *N* using a certificate contained in data string D_3, or by some other means (e.g. in a cache of such keys). Finally, *U* sends *N* message **M3**.
4. On receipt of **M3**, *N* verifies the signature of *U* and checks the correctness of *N*, R_N and R_U,, thereby verifying that **M3** was sent by *U* and was a reply to **M2**.

This protocol enables the user and the NO verify each other's identity. The certificate of user can be issued by the SP when the user is initially registered with the SP. The NO has to know the SP's valid public key. The certificate of NO should be issued by an entity whose valid public key must be known to the user before the authentication process starts.

Besides those discussed in subsection 5.3.1, other advantages of this protocol are:

1. The user secret key S_U is known only to the user, and the NO secret key S_N is known only to the NO.
2. The NO does not have to communicate with the SP in order to obtain the authentication information.
3. The NO does not have to maintain the confidentiality of the user public key.

5.3.7 Summary

In subsection 5.3 we have described four authentication protocols, one for each of the four role model options R1, R2, R3 and R4. Each of the authentication options can be implemented using protocols based on various cryptographic primitives (some or all of P1, P2, P3 and P4). It is important to note that the four protocols considered here just one of many choices for each role model option, and they are not necessarily the best choices. The selection of the protocols to be used in 3GS is a complex problem, and it very much depends on the exact requirements of 3GS for service related authentication. Some examples of considerations which will affect the final selection of protocol are as follows.

- What is an acceptable frequency for NO to SP communication?
- How much freedom is required for an SP to choose their own algorithm?
- Is it acceptable that the NO must preserve the confidentiality of authentication data?
- What is an acceptable figure for the amount of data which needs to be transferred on the radio path during the authentication process?
- What is the acceptable complexity for the authentication algorithms to be supported by 3GS, in particular by the user access device?

5.4 Service-related session key distribution

5.4.1 Introduction

This subsection discusses service-related session key distribution in third generation mobile telecommunications systems (3GS), which can be achieved during service-related authentication processes.

A definition of service related authentication in 3GS is given in subsection 5.2. We use the term service-related session key distribution to mean a procedure which occurs when a user accesses a 3GS service provided by a NO, and which provides:

- *session key distribution*, i.e. the user and NO agree upon a session key for subsequent use in protecting user/NO communications, and
- *session key confirmation*, i.e. the user and NO both know that each other has received the session key properly and believes in the suitability of the key.

The three roles identified in subsection 4.2 as being involved in service related authentication, are Users, Network Operators (NOs) and Service Providers (SPs). We classify session-key distribution protocols by the four role model options described in subsection 5.2, called R1-R4, based upon who can provide authentication information about the user. Each of these four options could be implemented using a number of different authentication protocols, and some possibilities for protocols are given in subsection 5.3.

As in the discussion of authentication protocols in subsection 5.3, for the purposes of this subsection key distribution mechanisms can be divided into the following three categories:

- **K1**. Key distribution using symmetric encryption techniques,
- **K2**. Key distribution using asymmetric cryptographic techniques, and
- **K3**. Key distribution using a cryptographic check function.

Examples of key distribution mechanisms in categories K1 and K2 have been defined in ISO/IEC 11770 parts 2 and 3 respectively, [5]. Mechanism category K3 has been analysed in a number of published papers, [41], [42], [43].

The session key to be established can be generated by any of: an SP, an old NO, a current NO, a user or both a current NO and user. Five options for service-related session key distribution, based on who can choose the session key, are as follows.

- **C1**. The SP chooses the session key.
- **C2**. The old NO chooses the session key.
- **C3**. The current NO chooses the session key.
- **C4**. The user chooses the session key.
- **C5**. The user and current NO both participate in choosing the session key.

Possible methods of key distribution of types K1, K2 and K3 will be described in subsection 5.4.2. Examples of service related session key distribution protocols with mutual authentication implementing R1, R2, R3 and R4 will be described and analysed in subsections 5.4.3, 5.4.4, 5.4.5 and 5.4.6 respectively.

For the same reason as in subsection 5.3, we consider only nonce-based protocols here. If third generation systems should require mechanisms based on time stamps or sequence numbers, then appropriate protocols can be found relatively easily.

5.4.2 Three kinds of key distribution mechanism

This subsection discusses possible methods of session key distribution using symmetric encryption techniques (K1), asymmetric cryptographic techniques (K2) and cryptographic check functions (K3). The general advantages and disadvantages of each category of mechanism can be found in subsection 5.3.2.

5.4.2.1 Key distribution using symmetric encryption techniques (K1)

In order to establish a session key secretly between two entities *A* and *B* using symmetric techniques, the session key exchanged within messages must be enciphered using shared secret keys. There are three basic sub-categories of key distribution mechanism using symmetric encipherment, namely: Point-to-Point, Key Distribution Centre (KDC) and Key Translation Centre (KTC) (see ISO/IEC 11770-2, [5]).

When *A* and *B* already share a key that can be used to establish further keys, a point-to-point environment exists. In this case, the session key can be generated by either *A* or *B* and then acquired by *B* or *A* correspondingly; alternatively the session key can be jointly generated by *A* and *B*, and computed and retrieved by both *A* and *B*.

If *A* and *B* wish to communicate with each other using only symmetric techniques but do not currently share such a key, they need to make use of a KDC or KTC.

The role of a KDC is to generate and distribute keys. In this case the session key is generated by the KDC and distributed to *A* and *B*.

The role of a KTC is to convert and distribute keys. In this case the session key is generated by either *A* or *B*, and distributed by the KTC to *B* or *A* correspondingly; alternatively the session key is jointly generated by *A* and *B*, and the respective components are converted and distributed by the KTC to *A* and *B*.

A number of examples of key distribution mechanisms in these three environments are defined in ISO/IEC 11770-2, [5].

5.4.2.2 Key distribution using asymmetric cryptographic techniques (K2)

Two fundamental methods for using asymmetric techniques to establish a session key between two entities *A* and *B* are described in ISO/IEC 11770-3, [5].

The first method is based on a structured setting which has to be shared by the two entities. The setting consists of the following objects: a set *G*, a set *H* and a function

$$F: H \times G \rightarrow G\,.$$

G, *H* and *F* must satisfy the following requirements:

1. *F* satisfies the commutativity condition

$$F(h, F(h', g)) = F(h', F(h, g)).$$

2. For almost all h_1 and h_2 in *H* it is computationally infeasible to find $F(h_1,F(h_2,g))$ from $F(h_1,g)$ and $F(h_2,g)$.

3. The entities *A* and B share a common element *g* in *G* which may be publicly known.

4. The entities acting on this setting can efficiently evaluate *F* for all *h* in *H*, *g* in *G*, and can efficiently generate random elements in *H*.

For instance, if *A* randomly generates h_A in *H* and sends $F(h_A,g)$ to *B*, and *B* also randomly generates h_B in *H* and sends $F(h_B,g)$ to *A*, *A* and *B* can compute the same session key as

$$K_{AB} = F(h_A, F(h_B, g)) = F(h_B, F(h_A, g))\,.$$

A number of examples of key agreement mechanisms using this method are discussed in ISO/IEC 11770-3, [5].

A second method is based on assuming that each entity X has a personal asymmetric encipherment system (e_X, d_X) and a personal asymmetric signature system (s_X, v_X), and that each entity has access to authenticated copies of the public encipherment and verification transformations of all other entities.

For example, A has obtained a key K and wants to transfer it securely to B. A encrypts K and some additional material using B's public encipherment transformation e_B, and then A signs this enciphered message using its private signature transformation s_A. After receiving a message containing, for example,

$$s_A(D_2 || e_B(K || D_1)),$$

B can uses A's public verification transformation v_A to verify the integrity and origin of this message; B then deciphers the message using B's private decipherment transformation d_B to obtain the key K. D_1 and D_2 are optional text fields, and, as elsewhere, $X||Y$ represents the concatenation of data strings X and Y in the order specified.

A number of examples of key transport mechanisms using this method are discussed in ISO/IEC 11770-3, [5].

5.4.2.3 Key distribution using a cryptographic check function (K3)

It is well known that if K is a random number, the function obtained by taking the exclusive-or of the one-way function f and K is also a one-way function. This idea is the basis of a number of possible key distribution mechanisms using cryptographic check functions.

Consider the following simple example. Suppose A and B share a secret key K_{AB} and a cryptographic check function f. When A generates a session key K_S and wants to transfer it securely to B, A can send B a message including

$$D || f_{K_{AB}}(D) \oplus K_S .$$

On receipt of this message, B can calculate

$$f_{K_{AB}}(D)$$

and then obtain K_S by exoring this value with the second part of the message received from A.

Based on the above idea, a number of similar examples using cryptographic check functions to distribute a session key have been proposed, [41], [42], [43].

5.4.3 A key distribution protocol for authentication option R1

We consider the situation where a user and an NO wish to communicate securely with each other, and for this purpose they need to verify the identity of one another and to establish a shared session key. In this role model option they do not currently share any secret or public information, and hence they must make use of an SP as an authentication server and a KDC or KTC.

As has been discussed in subsection 5.3, protocols in any of categories K1, K2 and K3 could be used. In the context of option R1:

- the obvious advantages of K1 are simpler algorithms, shorter transferred data and easier methods of session key distribution;
- the main advantage of K3 is that algorithms of K3 are simpler than K2 and might even be simpler than K1; moreover, and possibly most importantly, products incorporating cryptographic check functions are much more readily exported around the world than products implementing encryption functions, and are not subject to the strict controls applying to all implementations of encryption functions;

- the most serious flaw of K1 and K3 would appear to be the requirement for all relevant pairs of SPs and NOs to share a secret key which will result in potentially large complexity in key management;
- the main advantage of K2 is not to have to distribute a shared key between every relevant pair of NOs and SPs;
- the main disadvantage of K2 is that asymmetric algorithms tend to be much more computationally expensive.

Hence the choice of which protocol category to use to implement R1 must be made very carefully.

For option R1, the session key can be chosen by the SP (C1), the NO (C3), the user (C4) or both the user and NO (C5). We will now discuss in detail one example protocol, which uses symmetric encryption techniques (K1) and lets the SP choose the session key (C1).

This protocol is a modified version of the mechanism given in Section 6.3 of ISO/IEC 11770-2, [5].

In the context of this protocol, we assume that the user, NO and SP know the distinguishing names *U*, *N*, *S* of the user, NO and SP respectively. In addition the SP and user share a secret key K_{SU}; and the SP and NO share another secret key K_{SN}.

M1: $U \rightarrow N$: $U||N||R_U||D_1$

M2: $N \rightarrow S$: $U||N||R_U||R_N||D_2$

M3: $S \rightarrow N$: $D_5||e_{K_{SN}}(U||R_N||K_{UN}||D_4)||e_{K_{SU}}(N||R_U||K_{UN}||D_3)$

M4: $N \rightarrow U$: $D_7||e_{K_{SU}}(N||R_U||K_{UN}||D_3)||e_{K_{UN}}(U||R_U||R_N'||D_6)$

M5: $U \rightarrow N$: $D_9||e_{K_{UN}}(N||R_N'||R_U||D_8)$

where R_U, R_N and R'_N are three nonces generated by *U*, *N* and *N* respectively; D_i is the *i*th optional text field; K_{UN} is a session key for *U* and *N*; and e_K denotes encryption using the key *K*. The terminology defined here will be used throughout the report. Note that this is precisely the same protocol as described in subsection 5.3.3.

The procedural description of the protocol is as follows.

1. User *U* sends NO *N* a message **M1** containing the distinguishing names *U* and *N*, a nonce R_U, and, optionally, a text field *D1*.
2. On receipt of **M1**, NO *N* determines the appropriate SP *S* from some combination of *U* and the data string D_1, and then sends the SP *S* a message **M2** including the distinguishing names *U* and *N*, the nonces R_U and R_N, and, optionally, a text field D_2.
3. After receiving **M2**, the SP *S* chooses a secret authentication key K_{UN} for user *U* and NO *N*, then returns the NO *N* a message **M3** including a ticket for *U*, namely $e_{K_{SU}}(N||R_U||K_{UN}||D_3)$.
4. On receipt of **M3**, the NO *N* decrypts the first part using the key K_{SN}, and from checking the correctness of *U* and R_N, *N* verifies that message **M3** was from *S* and was a reply to **M2**. NO *N* then retrieves K_{UN}, and forwards the ticket concatenated with another message encrypted using K_{UN} to *U*.
5. On receipt of **M4**, user *U* decrypts the ticket using the key K_{SU}, and by checking the correctness of *N* and R_U, user *U* verifies the ticket was issued by *S* and was a reply to **M1**. User *U* retrieves the key K_{UN}, and then, after using K_{UN} to decrypt the other part of **M4** and checking the correctness of *U* and R_U, user *U* believes NO *N* has retrieved K_{UN}. User *U* then returns **M5** to NO *N*.

6. On receipt of **M5**, NO *N* decrypts **M5** using the key K_{UN}. By checking the correctness of *N*, R_U and R'_N, NO *N* verifies that **M5** was sent by user *U* and was a reply to **M4**. NO *N* can also be sure that *U* has retrieved the key K_{UN}.

This protocol enables the user and the NO to verify each other's identity, and, in addition, the user and the NO agree upon the secret authentication key K_{UN}, and the user and NO know that the key K_{UN} is possessed by each other. Optionally the key K_{UN} can be used as a session key for the provision of additional security features (e.g. data confidentiality) in subsequent communications between the user and the NO (see also subsection 5.4 below).

Besides those discussed in Section 5.3.1, other advantages of this protocol are:

1. The user specific key K_{SU} is known only to the user and SP; it does not have to be revealed to the NO. Similarly, the NO specific key K_{SN} is known only to the NO and SP; it does not have to be revealed to the user.

2. The SP keeps total control over the user's communications action because the SP chooses the session key.

Other disadvantages of this protocol are:

1. The NO has to communicate with the SP in order to obtain the authentication information.

2. The authentication algorithm has to be known to the NO, so it needs to be standardised.

3. The NO must preserve the confidentiality of the authentication key K_{UN}.

5.4.4 A key distribution protocol for authentication option R2

In this option, a user and the Network Operator NO_n do not currently share any secret or public information. When they wish to communicate securely with each other, they need to verify the identity of one another and to establish a shared session key between them. For this purpose they make use of an 'old NO' NO_o as an authentication server and a KDC or KTC. We use the abbreviations NO_o and NO_n for an old network operator (responsible for providing the previously visited network) and a new network operator (responsible for providing the present visited network) respectively.

Given that what is required is for two parties to authenticate one another with the aid of a third party, R2 is quite similar to R1. In fact, if we change SP into NO_o and NO into NO_n in the protocol described in subsection 5.4.3, we can get an example of a protocol implementing R2. As previously, protocols in any of categories K1, K2 and K3 can be used for the implementation of this option. In addition to the advantages and disadvantages described in subsection 5.3.2, one potential problem with protocols of types K1 and K3 is that all relevant pairs of NOs must share a secret key. As always, the main shortcoming of protocols of type K2 is the computational complexity of asymmetric algorithms.

The main difference from option R1 is that the 'authentication server and KDC or KTC' NO_o in R2 is not the party who issues the user's authentication information. The information about the user held by NO_o was issued by the SP at some previous time (perhaps using a protocol falling under option R1). In option R2, the session key can be chosen by the old NO (C2), the new NO (C3), the user (C4) or both the user and new NO (C5). We will now discuss in detail one example protocol, which uses a cryptographic check function (K3) and lets the old NO choose the session key (C2). If one changes *N'* into *S* in this protocol, it can be also used for implementing R1.

The following protocol is a session key distribution protocol with mutual authentication. In the protocol description below, *N* and *N'* represent the new NO and old NO's distinguishing names respectively. It is assumed that the NO_o and users share a secret key $K_{N'U}$, which could be a session key previously established (perhaps using a protocol implementing option R1). We also assume that NO_o and NO_n share a secret key, $K_{N'N}$.

M1: $U \rightarrow N: \ D_1 \| U \| N \| N' \| R_U$

M2: $N \rightarrow N' : \ D_2 || U || N || N' || R_U || R_N$

M3: $N' \rightarrow N :$

$$D_5 || R_{N'N} || c_{K_{NN'}}(R_N || R_{N'N} || U || D_4) \oplus K_{UN} || R_{N'U} || c_{K_{N'U}}(R_U || R_{N'U} || N || D_3) \oplus K_{UN}$$

M4: $N \rightarrow U :$

$$D_7 || R_{N'U} || c_{K_{N'U}}(R_U || R_{N'U} || N || D_3) \oplus K_{UN} || R'_N || c_{K_{UN}}(U || R_U || R'_N || D_6) \oplus K_{UN}$$

M5: $U \rightarrow N : \ D_9 || c_{K_{UN}}(N || R'_N || R_U || D_8)$

where c_K denotes a cryptographic check function using the key *K*.

The procedural description of this protocol is as follows:

1. *U* sends the distinguishing names *U*, *N* and *N'*, a nonce R_U and, optionally, a text field D_1 to *N*.
2. After receiving **M1**, *N* sends the same distinguishing names, two nonces R_U and R_N, and, optionally, a text field D_2 to *N'*.
3. On receipt of **M2**, *N'* generates two nonces $R_{N'N}$ (for *N*) and $R_{N'U}$ (for *U*) and a session key K_{UN}, and then sends *N* the message **M3**.
4. On receipt of **M3**, *N* uses the shared key $K_{N'N}$ to check of the first part of the message was issued by *N'*. *N* then retrieves the key K_{UN} and sends **M4** to *U*.
5. On receipt of **M4**, *U* uses the key $K_{N'U}$ in conjunction with the first part to recover K_{UN}. *U* can then check the second part of the message using K_{UN}, in order to verify that it was generated by *N* (and hence *N* must know K_{UN}). Finally *U* sends **M5** to *N*.
6. On receipt of **M5**, *N* can verify that *U* has retrieved K_{UN}.

This protocol enables the user and the NO to check each other's identity, to agree upon the session key K_{UN}, and to verify that the key K_{UN} is known to each other.

Disadvantages of this protocol include:

1. NO_n has to communicate with NO_o in order to obtain the authentication information.
2. Once a session key has been compromised, all subsequent session keys established using this protocol will also be compromised, because in this protocol the old session key is used to establish the new session key.

5.4.5 A key distribution protocol for authentication option R3

In this option, the user and NO already share the information necessary for authentication and key distribution. As previously, protocols in any of categories K1, K2 and K3 could be used for the implementation of this option. The session key in this option can be chosen by the NO (C3), the user (C4) or both the user and NO (C5). We now give one example protocol, which uses symmetric encryption techniques (K1) and in which the NO chooses the session key (C3).

The following protocol is based on the key establishment mechanism in Section 5.5 of ISO/IEC 11770-2, [5]. In this protocol it is assumed that the user and NO both know their respective distinguishing names *U* and *N*, and also a shared secret key K_{UN} (which might be a previous session key established using a different mechanism).

M1: $U \rightarrow N : \ D_1 || U || N || R_U$

M2: $N \rightarrow U : \ D_3 || e_{K_{UN}}(U || R_U || R_N || K'_{UN} || D_2)$

M3: $U \rightarrow N: \; D_5 \| e_{K'_{UN}}(N \| R_N \| R_U \| D_4)$

One difference between the above protocol and the mechanism in ISO/IEC 11770-2, [5], is that the session key K'_{UN} is generated by the NO and not by both the NO and the user.

The procedural description of this protocol is as follows:

1. *U* sends the distinguishing names *U* and *N*, a nonce R_U, and, optionally, a text field D_1 to *N*.
2. After receiving **M1**, *N* generates the session key K'_{UN} and sends **M2** to *U*.
3. On receipt of **M2**, *U* decrypts **M2** using the key K_{UN}, and, by checking the correctness of *U* and R_U, *U* is able to verify that **M2** was sent by *N* and was a reply to **M1**. *U* can next recover the session key K'_{UN} and, finally, *U* sends **M3** to *N*.
4. On receipt of **M3**, *N* decrypts **M3** using K'_{UN}, and, by checking the correctness of *N*, R_U and R_N, *N* verifies that **M3** was sent by *U* and was a reply to **M2**. Moreover *N* can also verify that *U* possesses the key K'_{UN}.

This protocol enables the user and the NO to verify each other's identity. In addition the user and NO agree upon the key K_{UN}, and verify that it is possessed by each other.

In addition to those discussed in subsection 5.3.1, other advantages of this protocol are:

1. The NO does not have to communicate with the SP in order to obtain the authentication information.
2. The session key is known only to the user and NO.

Other disadvantages of this protocol are:

1. The authentication algorithm has to be known to the NO, so it needs to be standardised.
2. The NO must preserve the confidentiality of the authentication key K_{UN}.
3. Once a session key is compromised, all subsequent session keys will also be compromised, because in this protocol the old session key is used to establish the new session key.

5.4.6 A key distribution protocol for authentication option R4

In this option, the authentication information about the user is provided by the user. Because the user and NO have no shared secrets, only cryptographic option K2 can be used here. The session key in this option can be chosen by the NO (C3), the user (C4) or both the user and NO (C5). In this subsection we consider an example protocol which uses asymmetric cryptographic techniques (K3) and allows both the user and NO participate in choosing the session key (C5). The protocol is taken from ISO/IEC 11770-3, [5].

M1: $U \rightarrow N: \; D_1 \| U \| N \| R_U$

M2: $N \rightarrow U: \; D_3 \| U \| R_U \| R_N \| F(h_N, g) \| s_{S_N}(U \| R_U \| R_N \| F(h_N, g) \| D_2)$

M3: $U \rightarrow N:$

$D_5 \| N \| R_N \| R_U \| F(h_U, g) \| s_{S_U}(N \| R_N \| R_U \| F(h_U, g) \| D_4) \| e_{K_{UN}}(U \| R_N \| D_6)$

M4. $N \rightarrow U: \; D_8 \| e_{K_{UN}}(N \| R_U \| D_7)$

where $s_K()$ denotes a signature computed using the key *K*, S_N denotes the secret signature key of *N*, S_U denotes the secret signature key of *U*, *F* is a function of the type described in subsection 5.4.2.2. Optionally, data strings D_3 and D_5 may contain certificates for the public keys of *N* and *U* respectively.

The recipient of such a certificate can use it to obtain a validated copy of the public key of the owner of the certificate.

The last part of message **M3** (namely $e_{K_{UN}}(U||R_N||D_6)$) and message **M4** are optional, and are present to provide key confirmation. The session key K_{UN} is computed by *U* and *N* respectively as

$$K_{UN} = F(h_U, F(h_N, g)), \text{ and}$$

$$K_{UN} = F(h_N, F(h_U, g)).$$

The procedural description of this protocol is as follows:

1. The user sends the distinguishing names *U* and *N*, a nonce R_U, and, optionally, a text field D_1 to *N*.

2. After receiving **M1**, *N* randomly and secretly generates h_N in *H*, and computes $F(h_N,g)$. Then *N* sends its signature on a data string including another nonce R_N and $F(h_N,g)$ to *U*.

3. On receipt of **M2**, *U* checks it by verifying the signature of *N* using a copy of *N*'s public signature verification key. This public key may either be obtained from a copy of *N*'s certificate or by some other means. *U* now retrieves $F(h_N,g)$, randomly and secretly generates h_U in *H*, and computes $F(h_U,g)$ and the session key

   $$K_{UN} = F(h_U, F(h_N, g)).$$

   Finally *N* sends its signature on a data string including $F(h_U,g)$, and, optionally, a key conformation message component to *N*.

4. On receipt of **M3**, *N* checks it by verifying the signature of *U*, retrieves $F(h_U,g)$, and then computes the session key

   $$K_{UN} = F(h_N, F(h_U, g)).$$

   Optionally, *N* decrypts

   $$e_{K_{UN}}(U||R_N||D_6)$$

   and checks the correctness of *U* and R_N, thereby verifying that *U* has correctly got this session key. Then, optionally, *N* sends another confirmation message (**M4**) to *U*.

5. On receipt of **M4**, after decrypting it and checking the correctness of *N* and R_U, *U* has verified that *N* has correctly got this session key.

This protocol allows the user and the NO to verify each other's identity. It also allows the user and the NO to agree upon the session key K_{UN}, and the user and NO can confirm that the key K_{UN} has been obtained by each other. The certificate of the user can be issued by the SP when the user is initially registered with the SP. In this case the NO would have to know the SP's valid public key. The certificate of the NO must be issued by an entity whose public key is reliably known to the user before the authentication and key distribution process starts.

Besides those discussed in Section 5.3, other advantages of this protocol are:

- The user secret key S_U is known only to the user, and the NO secret key S_N is known only to the NO.
- The NO does not have to communicate with the SP in order to obtain the authentication information and the session key.
- The NO does not have to maintain the confidentiality of the user public key.
- Neither the user nor the NO can predetermine the value of the session key.
- The session key is known only to the user and NO.

A disadvantage of this protocol is:

- To participate in choosing the session key could result in much increased complexity for the user access device.

5.4.7 Summary

In the above subsection we have described four session key distribution protocols with mutual authentication, which are used to implement authentication role model options R1, R2, R3 and R4. The four protocols considered here are fairly arbitrary choices from the many available. They are not necessarily the best choices. As has been mentioned in subsection 5.3, the selection of the protocols is complex, and it depends on the requirements of 3GS for service related authentication.

6. Multiple access methods and encryption of the air interface

6.1 Introduction

The choice of multiple access method for the third generation of mobile telecommunications systems has generated a lot of discussion. The purpose of this section is to assess the impact of this choice on the selection of mechanisms to provide encryption of user and other data on the air interface.

In subsection 6.2, the multiple access methods of possible relevance to third generation systems are surveyed, most importantly including TDMA and CDMA. Subsection 6.3 then contains a more detailed examination of encryption for TDMA systems, briefly reviewing how this is done in the GSM system (which uses TDMA for air interface multiple access). This leads naturally to subsection 6.4, where possibilities for combining encryption with CDMA are considered. This in turn raises a fundamental question, namely what sequences are appropriate for use in CDMA systems?

Subsection 6.5 attempts to provide an answer to this question. It compares the properties of random sequences with those of sequences believed to be 'good' for CDMA applications, and shows that, in some load conditions, random sequences may be superior. This suggests the radical conclusion that existing systems may be built on a possibly false premise, namely that CDMA sequences must be chosen with great care. This topic is the subject of continuing research.

6.2 Multiple access methods

There are a number of ways in which the bandwidth available to a system can be divided to accommodate a number of users wishing to use it.

In first-generation analogue systems, FDMA (frequency-division multiple access) is used. This means that the available bandwidth is divided into sub-bands, and each user who gains access to the system is assigned one unique sub-band for the entire duration of a call, after which the sub-band is relinquished and becomes available to any other user wishing to gain access to the system. In cellular systems, SDMA (space-division multiple access) is employed in addition. This just means that the same frequency is re-used in different cells, with the spatial separation sufficient to reduce interference to acceptable levels. Within each cell, FDMA ensures that there is insignificant interference between users.

6.2.1 Time-division techniques

FDMA/SDMA is the only technique available to analogue systems. Moving to digital systems allows other options. GSM uses a combination of FDMA and TDMA (time-division multiple access). This involves dividing the time axis into contiguous frames, each containing a predefined number of contiguous time slots. Each user of the system is assigned a unique time slot in each frame for the entire duration of a call. Once a call is terminated, the corresponding time slot is relinquished and becomes available to any other user wishing to gain access to the system. As before, SDMA allows users in different cells to use the same frequency.

Enhancements to TDMA have been proposed. These include:

- *DCA (Dynamic Channel Allocation)*

  In this scheme, which is used for the DECT air interface, higher bit-rates are achieved by assigning a number of channels to a particular call.

- *PRMA (Packet-Reservation Multiple Access)*

  This is a way of exploiting the natural pauses in speech. Essentially each terminal recognises the start of a ‘talk spurt’ and then competes for a timeslot with other terminals. Once it has a timeslot for the first speech packet, corresponding timeslots in subsequent frames are automatically reserved. However, unlike conventional TDMA, the timeslots are not reserved for the entire duration of the call but only for the duration of the ‘talk spurt’. In experiments, [44], it has been shown that this leads to a 50% increase in capacity, in reasonable conditions. RACE ATDMA have considered an enhanced version of PRMA, known as PRMA++.

  PRMA is essentially an adaptation of the Reservation ALOHA (R-ALOHA) data packet broadcast protocol for use in a cellular environment with speech transmissions, [45]. The principal differences between the two schemes are:

  - Acknowledgements in PRMA are from the cell base station to the mobiles and can thus be considered to have negligible delay compared with many applications of R-ALOHA (e.g. satellite broadcast networks).
  - Voice packets which are delayed more than a certain interval are dropped in PRMA due to the nature of speech transmission.

  Although PRMA is primarily a voice packet protocol, it can be adapted to accommodate both speech and random data transmissions (with priority given to voice packets), [46].

- *SFH (Slow Frequency Hopping*).

  Slow frequency hopping can optionally be used in a GSM network. When frequency hopping, consecutive TDMA bursts are transmitted at different frequencies according to a predefined hopping sequence. This provides resistance to Rayleigh fading since the signal nulls change in physical position as the frequency is changed. SFH also provides the advantage that co-channel interference is more evenly spread between users.

6.2.2 Spread-spectrum techniques

Spread spectrum techniques were originally developed and used in military applications for covert operation and to resist intentional jamming. There are two basic types of spread-spectrum system: frequency hopping (FH) spread spectrum systems and direct sequence (DS) spread spectrum systems.

We shall concentrate on DS systems. In DS systems, each bit of message data is effectively mapped to a codeword consisting of a sequence of bits known as *chips*. This process is known as *spread coding*. It is achieved by performing a multiplication between the message data and a relatively high rate spreading code (assuming all binary values to belong to {-1,1}). Because the chip rate of the spreading code is greater (usually much greater) than the bit rate of the message data, the effect is to spread the power of the message signal over a relatively wide bandwidth. The spreading may be such that the processed signal lies significantly below the level of background noise at a distant receiver, yet, in contrast to the case when conventional modulation schemes are employed, the signal is still recoverable. This facilitates covert operation. At the receiver, the same multiplication process is performed between the received signal and a locally generated and synchronised version of the spreading code. The effect now is to despread the wanted signal but spread any other received components including those due to intentional jamming. Because the wanted signal is now narrowband and at a high level relative to the other components which have been spread at the receiver, it can be resolved.

Both the aforementioned types of spread spectrum systems can be adapted for multi-user application. They are then known as frequency hopping spread spectrum multiple access (FH/SSMA) systems and direct sequence spread spectrum multiple access (DS/SSMA) systems respectively. Each of these can constitute a code-division multiple access (CDMA) system, although the term CDMA is usually associated with the latter and will be used in this sense in the following.

6.2.2.1 CDMA

In the most basic form of CDMA, each user is assigned a unique code for spreading/despreading purposes. This code is allocated for the whole duration of a call and then relinquished for use by any other user wishing to access the system. Two basic types of CDMA system may be identified:

- *Synchronous*: in which the bit and chip transitions of the various CDMA channels arriving at a receiver are guaranteed to be synchronised.
- *Asynchronous*: in which the bit and chip transitions of the various CDMA channels arriving at a receiver are not synchronised in general.

Synchronous CDMA systems lend themselves to *orthogonal* spreading in which each active user is assigned one code from a set of mutually orthogonal codes. During the modulation process, each bit of data pertaining to a particular user is replaced by the complete corresponding code (either directly or in inverted form depending upon the value of the data bit). In principle, this technique permits individual CDMA channels to be recovered at the receiver with zero cross-channel interference and without the use of power control.

Asynchronous CDMA systems are by far the more common. For such systems, the main operational requirements of the code set in use are, [47]:

- low autocorrelation coefficients for all code phases apart from those which are an integer multiple of the code length N (to permit synchronisation),
- low crosscorrelation coefficients for all code phases (to permit synchronisation and to ensure low cross channel interference).

In practice, additional correlation properties are required. This is because the data modulation superposed on the spreading sequences can potentially give rise to spurious correlation products in the receiver which would not be possible if the raw sequences themselves were being received. In such circumstances, it is necessary to distinguish between even (or periodic) and odd correlation functions, [48]. The former describe the correlation properties of unmodulated sequences and the latter those of modulated sequences. We shall concentrate on periodic correlation functions in this document.

In general, orthogonal spreading is not viable in asynchronous CDMA systems because orthogonal codes generally have poor autocorrelation and crosscorrelation properties. There is also no requirement for each data bit to be replaced by the complete spreading code during the modulation process. In fact, it is much more common for the complete code to be spread over several data bits. Because cross channel interference is non-zero in asynchronous CDMA systems, power control is essential.

The basic problems, as identified by Steele and Williams, [49], when implementing asynchronous CDMA systems, are:

- accurate power control,
- the generation of large families of codes with desirable correlation properties,
- synchronisation.

Methods of generating a large family of linear sequences with desirable correlation properties have been documented by Kasami, [50], and Gold, [51], [52] among others. See subsection 6.4.4 for more details.

6.2.2.2 Commercial/envisaged CDMA systems

6.2.2.2.1 The Qualcomm CDMA system

The solution that Qualcomm have proposed for a cellular system involves the base station (BS) transmitting a pilot signal on a common channel with a unique phase. This allows the MS to synchronise and identify the base station and to adjust the power of its transmissions. The system is synchronised using a GPS (Global Positioning System) signal, [53], [54].

Qualcomm have opted for a multi-level spread coding scheme. This is complicated and the codes are used asymmetrically on the forward and reverse links principally because the forward link constitutes a synchronous CDMA system whereas the reverse link constitutes an asynchronous CDMA system. Basically, there are three types of code: a short PN code (length 2^{15} chips), a long PN code (length 2^{42}-1 chips) and orthogonal codes based upon Walsh functions (length 64 chips). The short PN codes allow different base stations to be distinguished, the long PN codes provide security and scrambling and the orthogonal codes based upon Walsh functions differentiate between different user channels. The chip rate is fixed at 1.2288MChip/s, although data rates of 1.2kb/s, 2.4kb/s, 4.8kb/s and 9.6kb/s can be supported.

6.2.2.2.2 The RACE CODIT scheme

RACE 2020 CODIT have also proposed a CDMA scheme for use by a third-generation mobile system. This scheme is tailored to the requirements of UMTS. Three different chip-rates are proposed by CODIT. These are 1.023, 5.115 and 20 MChip/s. Each information bit rate offered by UMTS is mapped to one of these chip-rates.

6.2.2.2.3 Other CDMA schemes

Various hybrid schemes have been proposed, usually using TDMA with some elements of CDMA. Some were originally proposed for the GSM system.

6.2.3 The choice of multiple-access methods

6.2.3.1 UMTS

UMTS may well use a hybrid scheme. Indeed there is no reason why the multiple access scheme used on the uplink should be the same as that used on the downlink.

6.2.3.2 Broadband systems

It should be noted that mobile broadband systems, such as that being specified by the RACE MBS project, are unlikely to be able to use CDMA or FH/SSMA because of the vast chiprates involved. RACE MBS are planning to use bit-rates of around 100 MBit/s. Basically, each bit of data is turned into twenty or so bits of codeword. This requires a chip-rate of around 3 GChip/s, which is considered to be above the limit of

current technology. Also, a bandwidth of several GHz would be required, while only 1GHz is available (63-64 GHz).

6.3 Air-interface encryption in TDMA systems

6.3.1 GSM encryption

In the GSM system all digitised speech data and most signalling data are encrypted when sent over the air interface. Note that this does not constitute end-to-end encryption of the data. In particular, the Layer 1 data flow transmitted on the dedicated control channel (DCCH) and the traffic channel (TCH) is encrypted by the bit-by-bit addition of a bit stream generated by the ciphering algorithm A5.

The DCCH is only encrypted when an encryption key has been set. The key K_c used for the encryption is generated by both the mobile and the network. It is the output obtained from algorithm A8, by inputting the subscriber key K_i and the most recently generated RAND value. Hence there is a fresh key for each authenticated call.

On the DCCH and TCH, the encryption starts when the mobile terminal receives an instruction from the base station.

When a handover occurs, the necessary information is transferred by the network to the new base station, and the encryption continues using the same key. Synchronisation is achieved by using the TDMA frame structure and number. On input of a session key and a TDMA frame number, the algorithm produces a sequence of 114 bits for enciphering in the transmit slot and a separate sequence for enciphering in the receive slot.

Other factors influencing the choice of parameters for the algorithm include the speech coder, error propagation and delay.

6.3.2 Impact of TDMA enhancements on encryption

6.3.2.1 Dynamic channel allocation

For further study (see section 11).

6.3.2.2 Packet reservation multiple access

In PRMA, a user engaged in a speech conversation is allocated a new time slot each time a new ‘talk spurt’ is initiated (resources permitting). Thus, during a typical conversation, a user will transmit in a variety of time slots. The inherent security afforded by PRMA clearly depends upon the average rate at which time slots are reassigned. This in turn depends upon the sensitivity of the *voice activity detector* in the mobiles. For a *slow* detector, which regards silence as say a few seconds without detectable speech, several seconds and possibly minutes of conversation may be transmitted in the same time slot. For a *fast* detector, which regards inter- and possibly intra-word gaps as silence, the time slot assigned to a user may change several times during the course of a typical sentence.

Despite the obvious advantage of PRMA over TDMA from the perspective of inherent privacy, there are a number of potential weaknesses in the system. These arise as a consequence of the following important characteristic of the PRMA protocol. With analogy to the R-ALOHA protocol, at the end of each reverse link time slot, the base station broadcasts a feedback packet. In the case of successful interpretation by the base station of a transmission from a mobile, the packet is an acknowledgement containing the identity of the mobile with a reservation in that time slot for subsequent frames. In the case where either no transmission takes place from a mobile, or transmissions could not be interpreted by the base station (due to channel errors and/or collisions between two or more transmissions), the packet contains a *null* message to indicate that no reservation exists and hence the time slot is available for contention during the following frame. Consequently, security is potentially compromised for at least two reasons:

i. By listening to the broadcast feedback message, an interceptor can determine which mobiles are transmitting and on which time slots. It is thus vital that some form of protection is afforded to this message. Encryption and/or the use of temporary mobile subscriber identities (which may need to be updated several times during a call) are possible solutions.

ii. When a time slot is relinquished, it cannot be used during the following frame because the base station must detect that no transmission is taking place on a time slot before it can inform all mobiles of the

availability of that time slot. This gap in transmissions aids an interceptor in establishing on which time slots new 'talk spurts' are appearing. Depending upon the dynamics of the voice activity detector, it may be possible to modify the protocol such that a mobile informs the base station in real time that it is transmitting the last packet of a 'talk spurt'. This will remove the requirement for gaps in transmissions on timeslots (and improve channel efficiency).

6.3.2.3 Slow frequency hopping

For further study (see section 11).

6.4 Air-interface encryption in CDMA systems

6.4.1 General remarks

6.4.1.1 High chip rates

The high chip rates proposed (up to 20 MChip/s for CODIT) may make it difficult to encrypt on a chip-by-chip basis. However, it is clear that GSM-style encryption would have to be performed after channel coding but before the spread coding, in order that the codewords could be detected by the receiver. Hence, encryption could be performed on data at the information bit rate.

6.4.1.2 Unsynchronised transmission

The CDMA scheme proposed by Qualcomm is synchronised using a GPS signal. However, unsynchronised CDMA schemes have been proposed, for instance by CODIT. The potential difficulties for encryption are clear.

6.4.1.3 Combining spread coding and encryption

There is a possibility that the selection of PN sequences could be made a cryptographic process. That is, the selection of a sequence for each user depends on a secret key shared by the base station and mobile station. The base station has to ensure that the sequence generated is still unique to the user, but an eavesdropper is unable to determine which sequence is being used by a user. This may require strict synchronisation with the base station.

6.4.2 Inherent security provided by a CDMA system

6.4.2.1 Discussion

CDMA systems are often described as affording a high level of security by their very nature, although such statements are rarely qualified. Hence, before deciding upon whether or not explicit encryption is required for CDMA based systems, the level of inherent security afforded by such systems must be examined in detail. Methods of making CDMA systems intrinsically secure also need to be investigated.

The privacy afforded by an unencrypted CDMA based multiple access system depends upon:

i. The secrecy of the set of spreading codes.

ii. The secrecy of the synchronisation information needed for despreading individual transmissions at the receiver.

iii. The covertness of the transmission i.e. whether or not transmissions are below the background noise level as seen at the receiver.

In commercial applications, the advantage afforded by covert operation will often be lost to some extent because the transmission bands will be fixed by regulatory bodies. Thus, although an interceptor may not know *a priori* if a transmission is taking place, he will know the exact frequency band of the transmission should it be taking place. Furthermore, should the interceptor be sufficiently close to the transmitter, the S/N ratio may be such that direct demodulation of the chips of the spreading sequence may be feasible. *This is an extremely serious security issue*. Should it be possible to algebraically determine the complete spreading code from directly demodulating only a part of the code, the system is inherently insecure.

In any case, i. and ii. suggest that security can be enhanced by:

i. Periodically or randomly switching between several sets of spreading codes (*code hopping*),

ii. Periodically or randomly introducing random phase shifts in the spreading codes (*phase hopping*),

both of which require strict synchronisation between transmitter and receiver. Do these methods affect the functionality or operational requirements of the CDMA based multiple access system (apart from increasing the complexity of the hardware/software at the transmitter and receiver)?

If the set of spreading codes is unknown, the difficulty in deducing individual codes may/will depend upon:

i. The length of the codes.

ii. The nature of the codes (sequences generated by non-linear means being more secure than sequences of the same length generated by linear means, [55]).

iii. The secrecy of the length of the codes and how they are generated.

iv. The degree to which the codes resemble purely random sequences, [55].

v. The number of users accessing the system (assuming that the number of codes transmitted is equal to the number of users accessing the system).

vi. The uncertainty in the relation between the bit and chip rates (if a large number of bit rates can be mapped onto each chip rate, the system may/will be more secure).

vii. The security of the synchronisation process (most methods of achieving synchronisation between transmitter and receiver rely on the spreading code being directly transmitted i.e. with no data modulation)

Note that the difficulty in deducing a keystream sequence in a stream cipher system also depends upon i., ii., iii. and iv. This analogy is not surprising in view of the fact that both spread spectrum and stream cipher systems involve performing a modulo-2 addition between data and a pseudo noise sequence.

With respect to v., is the system more secure if all codes are transmitted irrespective of whether there is a user transmission to support each? This will undoubtedly increase the level of self-interference of the system.

Is it necessary for an interceptor to deduce the exact codes? Will a high level of correlation be sufficient to make speech transmissions intelligible?

6.4.2.2 Conclusion

The worst case under normal operating conditions is clearly that in which only one user is actively using the system and an eavesdropper is situated close to both the user and the base station. This scenario should therefore receive the most attention. Certainly, any CDMA system which is proposed as being inherently secure must be secure under this condition.

6.4.3 Code families and their properties for synchronous CDMA systems

6.4.3.1 Walsh codes

A set of N Walsh codes of length N exists for each N which is an integer power of 2. These N Walsh codes form the rows (or equivalently columns) of an $N \times N$ Hadamard matrix. Hence, each of the N Walsh codes is orthogonal to each of the other (N-1) Walsh codes from the set. However, for some shifts, autocorrelation and crosscorrelation coefficients can be extremely large in absolute terms (even as large as N, the autocorrelation coefficient corresponding to zero shift), [47].

The implication of the above discussion is that Walsh codes are only useful as spreading codes in synchronous CDMA systems for which, at a receiver, the bit and chip transitions of all CDMA channels coincide with each other. As an example, Walsh codes are employed as spreading codes in the forward (base station-user terminal) direction of the Qualcomm CDMA system. At any given user terminal, all CDMA channels are received with bit and chip transition alignment. Walsh codes are useless as spreading codes in asynchronous CDMA systems because of their undesirable correlation properties.

The security afforded by Walsh codes is very small. Most do not resemble random sequences at all (this explains why certain autocorrelation and crosscorrelation coefficients can be relatively large in an absolute sense). Such characteristics are shared by most large families of orthogonal codes. Hence, if the desirable characteristics afforded by orthogonal codes when used in CDMA systems (zero cross channel interference, no power control) are required, explicit encryption must be undertaken before spreading takes place.

6.4.4 Code families and their properties for asynchronous CDMA systems

This subsection concentrates on certain families of codes which have been proposed for use as spreading codes in asynchronous CDMA systems. Particular attention is given to the correlation properties, set sizes and linear complexity of these code families. The set of code families discussed is of course not exhaustive and will be updated as more literature becomes available.

6.4.4.1 m-sequences

A linear feedback shift register (LFSR) with n stages can generate sequences with length up to $N = 2^n-1$ bits. A sequence with the maximum possible length is known as an *m*-sequence. For an *m*-sequence to be generated, the LFSR must have feedback connections which correspond to a primitive polynomial and its initial state (i.e. the initial contents of the shift register) must be non-zero, [47], [56].

The periodic autocorrelation function of an m-sequence of length $N = 2^n-1$ is given by, [47], [56]:

N for a shift which is an integer multiple of N,

-1 for all other shifts,

which, for large values of N, is a reasonable approximation to the periodic autocorrelation function of a purely random sequence of the same length.

In spite of their desirable autocorrelation correlation properties and ease of generation, the use of m-sequences as spreading codes in CDMA systems is disadvantageous for two reasons, [47], [56]:

i. The number of m-sequences of a given length is relatively low (particularly if n is even), being given by $\phi(N)/n$ where ϕ is the Euler function.

ii. If just $2n$ consecutive bits of the m-sequence are known, the whole m-sequence can be determined via the Berlekamp-Massey algorithm.

6.4.4.2 Gold codes

Gold discovered that a selected pair of m-sequences of length $N = 2^n-1$ exhibit a three valued periodic crosscorrelation function with a reduced upper bound on the correlation levels as compared with the rest of the m-sequence set of the same length, [47], [51], [52]. This pair of m-sequences is referred to as the *preferred pair* and is unique for a given sequence length. For the preferred pair of m-sequences of length N, the periodic autocorrelation (for any shift which is not an integer multiple of N) and crosscorrelation levels are restricted to the three values, [47]:

$$\{-t(n), -1, t(n)-2\}$$

in which:

$$t(n) = 2^{(n+1)/2} + 1 \quad \text{for } n \text{ odd},$$

$$t(n) = 2^{(n+2)/2} + 1 \quad \text{for } n \text{ even}.$$

Gold has presented the method for the selection of the preferred pair of m-sequences from the set of m-sequences of a given length, [51], [52].

The enhanced correlation properties of the preferred pair of m-sequences can be inherited by any sequence derived from the original pair by a process of modulo-2 addition. There are a possible 2^n-1 relative phase shifts between the preferred pair of m-sequences of length 2^n-1, and hence 2^n-1 possible sequences can be generated by this process of modulo-2 addition. These derived sequences, together with the original preferred pair of m-sequences, constitute the family of Gold codes of length $N=2^n-1$. Thus, there are $N+2 = 2^n+1$ Gold codes of length $N = 2^n-1$, [47].

Gold codes have several advantages over *m*-sequences as spreading codes in asynchronous CDMA systems for a given sequence length, [47]:

i. Periodic crosscorrelation properties are on average better.

ii. A much larger number of codes are available (particularly if *n* is even).

iii. Security is slightly improved (i.e. more than 2*n* consecutive bits will be required to characterise a Gold code unambiguously).

On the other hand, those Gold codes which are not *m*-sequences have worse periodic autocorrelation properties than m-sequences of the same length. There is also a slight implementation cost. This applies whether the Gold codes are generated as previously described, with the output of two maximal length linear feedback shift registers (with characteristic polynomials corresponding to the preferred pair of m-sequences) being subject to modulo-2 addition, or a single linear feedback shift register (with characteristic polynomial equal to the modulo-2 product of the characteristic polynomials corresponding to the preferred pair of m-sequences) is employed to generate the Gold codes, [47].

6.4.4.3 Small set Kasami codes

The generation of small set Kasami codes is similar to that of Gold codes in that it involves (conceptually at least) modulo-2 addition of preferred pairs of m-sequences, [47], [50]. This time, however, one m-sequence of the preferred pair evolves from a maximal length linear feedback shift register with double the number of stages of that of the other. Thus, one m-sequence from the preferred pair has length 2^{2n}-1 while the other has length 2^n-1.

Kasami has presented the method for the selection of the preferred pair of *m*-sequences, [50]. This involves correctly selecting the longer m-sequence and decimating it to yield the smaller *m*-sequence.

There are a possible 2^n-1 relative phase shifts between a preferred pair of *m*-sequences of length 2^{2n}-1 and 2^n-1, and hence 2^n-1 possible sequences can be generated by this process of modulo-2 addition. These derived sequences, together with the original *m*-sequence from the preferred pair of length 2^{2n}-1, constitute the family of small set Kasami codes of length $N = 2^{2n}$-1. Thus, there are 2^n small set Kasami codes of length $N = 2^{2n}$-1, [47].

For small set Kasami codes of length $N = 2^{2n}$-1, the periodic autocorrelation (for any shift which is not an integer multiple of *N*) and crosscorrelation levels are restricted to the three values, [47]:

$$\{-t(n),-1,t(n)-2\}$$

in which:

$$t(n)=2^n+1.$$

Both the autocorrelation and crosscorrelation properties of small set Kasami codes are superior to those of Gold codes of the same length. However, the number of small set Kasami codes which can be generated is much lower than the number of Gold codes which can be generated for the same sequence length (and is generally only slightly greater than the number of m-sequences which can be generated for the same sequence length), [47].

6.4.4.4 Bent function sequences

Bent function sequences are very similar to small set Kasami sequences of the same length in respect of their correlation properties and set sizes, [47], [57]. However, due to the non-linear manner in which they are generated, the linear complexity of and hence security afforded by these sequences is far superior to small set Kasami sequences of the same length. The principal problem in employing Bent function sequences as spreading codes in secure CDMA systems is the associated implementation cost and effort: they are considerably more difficult to generate than small set Kasami sequences of the same length.

6.4.4.5 Interleaved sequences

Families of non-linear sequences can be generated by interleaving m-sequences according to some predefined algorithm.

A class of *m-like sequences* has been reported in the literature, [58]. These are generated by interleaving certain m-sequences with null sequences in a particular order. Their usefulness arises from the fact that they have the same periodic autocorrelation function as an m-sequence of the same length, but a much higher linear complexity.

A class of *Kasami-like sequences* has also been reported, [59]. These are generated by interleaving certain m-sequences with the interleaving order governed by a set of sequences called *prime sequences*. Their usefulness arises from the fact that they have the same periodic autocorrelation and crosscorrelation functions as a small set Kasami sequence or Bent function sequence of the same length, but a higher linear complexity. In addition, for a given sequence length, more Kasami-like sequences than true Kasami sequences are available.

6.4.4.6 Geometric sequences

Geometric sequences are a very general class of pseudo random binary sequences which include the m-sequences and the GMW sequences as special cases, [60].

The subset of geometric sequences constructed over *GF(p)*, where *p* is an odd prime, are of particular interest as they can be shown to possess an extremely high linear complexity over *GF(2)*. Unfortunately, the linear complexity relative to the source field *GF(p)* is low, [61], and the periodic autocorrelation coefficients of these sequences at certain non-zero shifts are unacceptably high, [60].

The general method of generating a geometric sequence is to apply a non-linear feedforward function to the output of a maximal period linear feedback shift register. This is a relatively simple task in practice, [60].

6.4.4.7 GMW sequences

GMW (Gordon-Mills-Welch) sequences have the same periodic autocorrelation function as m-sequences of the same length. Their crosscorrelation functions comprise three relatively low values. However, the linear complexity of GMW sequences, although significantly higher than that of m-sequences/Gold codes/small set Kasami Codes of the same length, is not considered to be sufficient for secure communications, [60].

6.4.4.8 Cascaded GMW sequences

Cascaded GMW sequences share many of the desirable properties of both the geometric sequences and the GMW sequences. By analogy to GMW sequences, cascaded GMW sequences have the same periodic autocorrelation function as m-sequences of the same length and their crosscorrelation functions comprise three relatively low values. The linear complexity of cascaded GMW sequences increases exponentially with the number of steps in the cascade, [60].

As far as we understand the concept of cascaded GMW sequences, the ‘cascade’ refers to a ‘tower’ of GMW-type functions in the non-linear feedforward function (refer to the section on geometric sequences for an explanation of the relevance of the feedforward function).

6.4.4.9 Blum sequences

In contrast to the majority of sequences considered so far, Blum sequences are based upon arithmetic over finite rings rather than finite fields. The basis of Blum sequences is therefore the same as that of some public-key ciphers and hence, if carefully designed, Blum sequences may offer inherent cryptographic security, [62], [63], [64].

The Blum or x^2 *mod n* generator is fundamentally different from the FSR generators which are employed to implement the majority of sequences considered so far, [62], [63], [64]. The generator uses a modulus *n* which is a strong Blum integer (i.e. the product of two strong primes each congruent to 3 *mod* 4) and a seed value x_0. The seed is successively squared *mod n* and the binary equivalent of the result forms the next $log_2 n$ bits of the Blum sequence. Such a sequence has been shown to exhibit a very high degree of randomness - approaching that of the Vernam one- time pad, [62], [63].

Blum sequences also offer other potential advantages over sequences constructed over finite fields for use in CDMA systems, [64]:

- A large number of Blum sequences of a given length and with desirable correlation properties may be available due to the fact that a finite ring is less structured than a finite field.
- Operations in a finite ring are simpler than those in a finite field due to the Chinese Remainder Theorem.

Despite these advantages, certain characteristics of Blum sequences need to be investigated to evaluate their feasibility for use in CDMA systems, [64]:

- Their autocorrelation and crosscorrelation properties.
- The ability to achieve rapid synchronisation between transmitter and receiver. This may not be a significant problem if the philosophy behind the Qualcomm CDMA system is adopted i.e. to synchronise to a relatively short (insecure) pilot PN code to which the relatively long secure spreading code has a secret phase relationship known only to the transmitter and receiver.
- The ease with which suitable values of the modulus n and the seed x_0 can be determined.

6.4.4.10 Other sequences

Other families of sequences which might be considered are the polyphase sequences, Kerdock codes and Preparata codes.

6.5 Security versus correlation properties of spreading sequences for CDMA

6.5.1 Introduction

Part of the text in subsections 6.5.3 and 6.5.4 has been published, [65], under the auspices of the 3GS3 project.

This subsection begins with an examination of the types of correlation functions which influence the performance of asynchronous code division multiple access (CDMA) systems i.e. those CDMA systems in which the bit and chip transitions of the various channels are *not* guaranteed to be time aligned (synchronised) with each other at an arbitrary receiver. The types of correlation function which influence the performance of synchronous CDMA systems are a subset of these. Correlation functions are so vital in CDMA systems because, correctly interpreted, they yield an indication of the level of co-channel interference which is likely to be experienced between users of the system and also of the likelihood of false synchronisation.

It will be seen that, because of certain inherent aspects of asynchronous CDMA systems, it is generally very difficult to mathematically analyse the correlation properties of spreading codes in a way which accurately reflects the level of co-channel interference which exists between users *under realistic operating conditions*; simulation must instead be employed. We simply analyse spreading codes in terms of the basic correlation functions.

Codes such as Gold codes and small set Kasami codes which are often employed as spreading codes in asynchronous CDMA systems have desirable correlation properties but low linear complexity. Thus, it is a relatively simple matter to determine the complete spreading code from an examination of only a relatively small contiguous part (certainly significantly less than the period) of any such code.

In contrast, cryptographically secure codes, although obviously generated deterministically, exhibit a near perfect randomness property in that, given a contiguous section of the sequence which is less than its period, it is almost infeasible to predict the next bit in the sequence. In the limit, then, the correlation functions of these codes must approach those of purely random sequences of the same 'period'.

A major part of this subsection is therefore devoted to comparing the correlation functions of purely random sequences with those of types of deterministic sequences which are commonly employed in asynchronous CDMA systems. The aim is to determine whether a compromise may be required between the security and correlation (interference) properties of spreading codes for asynchronous CDMA systems or whether the best spreading code is simply that which exhibits the highest degree of randomness.

6.5.2 Types of correlation function

6.5.2.1 True correlation functions

In general, several types of correlation function influence the performance of asynchronous code division multiple access (CDMA) systems, [66,67]. These different types will be explained in connection with the cross-correlation which exists between two sequences $\{a_k\}$ and $\{b_k\}$ of length n chips $(k=0,1,2,...,n-1)$. Analogous definitions for the autocorrelation of a single sequence follow by assigning $\{a_k\} = \{b_k\}$.

Of particular interest are the *even* (or *periodic*) and *odd* cross-correlation functions. The former represents the cross-correlation function of the two sequences considered in their entirety if both are unmodulated by data. The latter represents the cross-correlation function of the two sequences considered in their entirety if one is modulated by a ...1010... bit stream (with each data bit being replaced by the complete sequence of n chips) and the other is unmodulated by data. These two types of cross-correlation function correspond to the extremes of those which may be encountered in practice in which, at the CDMA receiver, a received sequence modulated by unknown data is correlated with a locally generated version of another sequence which is unmodulated (assuming that each bit of data is replaced by the complete sequence of n chips).

Both the even and odd cross-correlation functions are examples of circular correlation functions i.e. the sequences are assumed to repeat continuously (as would be the case in a CDMA system). Hence the correlation value for a particular relative delay τ between the two sequences is evaluated as the summation of the bitwise products over a series of n contiguous chips (all bit and chip values assumed to belong to {+1,-1}).

It is also useful to introduce another type of cross-correlation function, the *aperiodic* cross-correlation function. This is not a circular correlation function i.e. only one instance of each sequence is considered. Hence the correlation value for a particular relative delay τ between the two sequences only considers the 'overlap' between them and is evaluated as the summation of the bitwise products over a series of n-τ contiguous chips (all bit and chip values assumed to belong to {+1,-1}). As with the even (or periodic) cross-correlation function, the effect of data modulation is not considered in the definition of the aperiodic cross-correlation function. The aperiodic cross-correlation function is the basic cross-correlation measure for asynchronous CDMA systems because it can be shown [68] that the performance of such systems (in terms of the average S/N ratio at the output of a conventional matched filter receiver) depends exclusively on the aperiodic and not the even or odd cross-correlation functions.

The following illustrations should clarify the definitions of the even (or periodic), odd and aperiodic cross-correlation functions. Note that we assume binary sequences with values belonging to {+1,-1} throughout for simplicity. A generalised discussion applicable to complex valued sequences can be found in [66] and [67].

The Even (or Periodic) Cross-Correlation Function $\theta_{a,b}(\tau)$:

Relative delay = τ $\qquad a_k, b_k \in \{+1,-1\}$

$$\theta_{a,b}(\tau) = \sum_{i=0}^{n-\tau-1} a_i b_{i+\tau} + \sum_{j=n-\tau}^{n-1} a_j b_{j-n+\tau}$$

The Odd Cross-Correlation Function $\hat{\theta}_{a,b}(\tau)$:

Relative delay = τ $\qquad a_k, b_k \in \{+1,-1\}$

$$\hat{\theta}_{a,b}(\tau) = \sum_{i=0}^{n-\tau-1} a_i b_{i+\tau} - \sum_{j=n-\tau}^{n-1} a_j b_{j-n+\tau}$$

The Aperiodic Cross-Correlation Function $C_{a,b}(\tau)$:

Relative delay = τ $\qquad a_k, b_k \in \{+1,-1\}$

$$C_{a,b}(\tau) = \sum_{i=0}^{n-\tau-1} a_i b_{i+\tau}$$

The even and odd cross-correlation functions may clearly both be defined in terms of the aperiodic cross-correlation function [66,67]. Specifically:

$$\theta_{a,b}(\tau) = C_{a,b}(\tau) + C_{a,b}(\tau - n)$$

and

$$\hat{\theta}_{a,b}(\tau) = C_{a,b}(\tau) - C_{a,b}(\tau - n)$$

6.5.2.2 Partial correlation functions

The even and odd correlation functions described in subsection 6.5.2.1 may be employed to characterise the performance of a CDMA system in which each bit of data in an arbitrary channel modulates (selectively inverts) the same sequence of *n* chips, i.e. in which the entire spreading code replaces each bit of data, either in straight or inverted form.

Unfortunately, most asynchronous CDMA systems do not work on this principle. The entire spreading code tends to be spread over several (possibly thousands) of data bits. However, at the receiver, decisions are still made on a bit-by-bit basis. Hence, the emphasis switches to correlation functions of sub-sequences of the spreading codes, i.e. partial correlation functions. If, for spreading codes of period *n* chips, only n_b chips are employed to spread each bit of data, then partial correlation functions need to be evaluated for subsequences of length n_b chips. Clearly, both even and odd partial correlation functions must still be considered in order to take account of the effect of data modulation. Consider two spreading codes $\{a_k\}$ and $\{b_k\}$ of length *n* chips ($k=0,1,2,...,n-1$) which comprise subsequences of length n_b chips thus:

$$\{a_k\} = \{\{a_{1,p}\}||\{a_{2,p}\}||...\}$$

where || denotes concatenation of sequences, and

$$\{b_k\} = \{\{b_{1,p}\}||\{b_{2,p}\}||...\}$$

where *p* may take values 0,1,2,...,n_b-1.

Even and odd partial cross-correlation functions are defined as follows:

Even (or Periodic) Partial Cross-Correlation Function $\theta_{a_l,b_m}(\tau)$:

Relative delay = τ $\qquad a_{l,p}, b_{m,p} \in \{+1,-1\}$

$$\theta_{a_l,b_m}(\tau) = \sum_{i=0}^{n_b-\tau-1} a_{l,i} b_{m,i+\tau} + \sum_{j=n_b-\tau}^{n_b-1} a_{l,j} b_{m+1,j-n_b+\tau}$$

Odd Partial Cross-Correlation Function $\hat{\theta}_{a_l,b_m}(\tau)$:

Relative delay = τ $\qquad a_{l,p}, b_{m,p} \in \{+1,-1\}$

$$\hat{\theta}_{a_l,b_m}(\tau) = \sum_{i=0}^{n_b-\tau-1} a_{l,i} b_{m,i+\tau} - \sum_{j=n_b-\tau}^{n_b-1} a_{l,j} b_{m+1,j-n_b+\tau}$$

Because the number of subsequences of length n_b chips which make up each spreading code of length *n* chips will in general be relatively large, the number of partial correlation functions which need to be evaluated can be extremely large.

6.5.2.3 Other correlation functions

Thus far, we have only considered correlation functions pertaining to two sequences. However, at a conventional CDMA receiver, the signals corresponding to all CDMA channels will be correlated with a locally generated version of the spreading code of the wanted channel. This is another complication in the analysis of the performance of CDMA systems although we shall not consider it further.

6.5.3 Correlation functions of purely random sequences

Consider two sequences $\{a_k\}$ and $\{b_k\}$ of length n chips ($k=0,1,2,...,n-1$) which are generated purely at random so that each chip of each sequence is independent of the other ($n-1$) chips in the sequence. Now, suppose that such sequences are employed as spreading codes in a CDMA system. This requires that each sequence is repeated indefinitely.

Clearly, a binary sequence of length n chips generated purely at random can assume any one of 2^n configurations. This implies that, when working with such sequences, each correlation coefficient pertaining to a particular type of correlation function has a distribution of possible values rather than a single value.

From a security perspective, it is clearly optimum that $P(0)=P(1)=1/2$ holds for the spreading sequence since then the probability of generating any one of the 2^n possible configurations is equal to that of generating any of the others (i.e. we have maximum entropy). The following discussion and analysis is specific to this case. We shall highlight those facts which are dependent upon the assumption that $P(0)=P(1)=1/2$.

The following facts are apparent:

- The distribution of autocorrelation and cross-correlation coefficients of purely random sequences of a given 'period' are identical.
- The distribution of even and odd correlation coefficients of purely random sequences of a given 'period' are identical (but note only for $P(0)=P(1)=1/2$).
- Whether the complete spreading sequence replaces each bit of data or is spread over several bits of data is immaterial as far as the distribution of correlation coefficients is concerned. The correlation coefficients are evaluated through consideration of a sequence of n_b chips per data bit.

Now, the even or odd correlation coefficient for a particular relative delay τ between two sequences of length n_b chips, which we shall call $\rho(\tau)$ to be generic, is evaluated as the summation of bitwise products over a series of n_b contiguous chips (all chip values are assumed to belong to {+1,-1}). A "+1" occurs where the two original bits were the same, and a "-1" where they were different. Hence, the value of $\rho(\tau)$ reduces to the number of agreements $A(\tau)$ minus the number of disagreements $D(\tau)$ for this value of τ. Thus:

$$\rho(\tau) = A(\tau) - D(\tau) = A(\tau) - (n_b - A(\tau)) = 2A(\tau) - n_b$$

Now, the bitwise product of two sequences of length n_b, each of which is generated purely at random with each bit such that $P(0)=P(1)=1/2$, is another purely random sequence of the same length also with the property that $P(0)=P(1)=1/2$. Hence, the number of agreements $A(\tau)$ follows a binomial probability distribution:

$$A(\tau) \sim B\left(n_b, \frac{1}{2}\right).$$

The two most important moments for $\rho(\tau)$ are thus given as

$$E\{\rho(\tau)\} = 2E\{A(\tau)\} - n_b = 2(n_b/2) - n_b = 0$$

and

$$Var\{\rho(\tau)\} = 2^2 Var\{A(\tau)\} = 4(n_b/4) = n_b.$$

For $n_b \geq 10$, the distribution of $A(\tau)$ and hence $\rho(\tau)$ can be approximated by a normal distribution. Hence, in such cases:

$$\rho(\tau) \sim N(0, n_b).$$

Unfortunately, successive correlation coefficients (i.e. $\rho(\tau)$ for different values of τ) are not independent of each other and this complicates any analysis of the extent of co-channel interference which may be suffered by users of a hypothetical CDMA system in which the spreading codes are purely random sequences.

6.5.4 Comparison of cross-correlation properties

6.5.4.1 Introduction

In this subsection, we shall be principally concerned with comparing the cross-correlation properties of purely random sequences with those of Gold codes, the latter being a class of deterministic pseudo-random sequences often employed as spreading codes in asynchronous CDMA systems [67].

Traditionally, the cross-correlation properties of code families for asynchronous CDMA systems have been assessed purely on the basis of the maximum absolute even (periodic) cross-correlation between any two members of the family. There are several reasons why this approach is not entirely satisfactory:

- Even and odd cross-correlation values occur with approximately equal frequency in an asynchronous CDMA receiver, and therefore both types should be considered.
- Because the entire code may be spread over several data bits, partial rather than true cross-correlation products may be more important.
- Maximum absolute cross-correlation values are related to worst case system performance, whereas it may be more meaningful to consider some averaged cross-correlation value which will be indicative of average system performance.
- The average performance of asynchronous CDMA systems depends exclusively on the aperiodic and not the even cross-correlation function [68].

For simplicity, we take the following approach. It is assumed, unrealistically, that each bit of data is replaced by a complete Gold code rather than the latter being spread across several bits of data. Furthermore, the effects of data modulation will not be considered with respect to the correlation properties of Gold codes (with reference to subsection 6.5.3, the effect of data modulation is immaterial with respect to the correlation properties of purely random sequences).

Thus, we shall be considering explicitly only true (as opposed to partial) even (periodic) cross-correlation functions. While this is a severe restriction, we can still address the third bullet point raised above. Specifically, comparisons will be made on the basis of:

- the mean square periodic cross-correlation coefficient value, and
- the probability that a given absolute periodic cross-correlation threshold is exceeded.

The former method provides a useful summary of the relative merits (with respect to correlation properties) of the two sequences. The latter is more complicated but also more useful as co-channel interference can usually be tolerated to a certain extent within a CDMA system; only when a given threshold level is exceeded will significant problems start to occur.

6.5.4.2 Comparisons based upon mean square correlation values

6.5.4.2.1 Introduction

As discussed in subsection 6.5.4.1, the mean square cross-correlation pertaining to a particular code family affords an indication of the average performance of an asynchronous CDMA system employing

this particular code family as a set of spreading codes. For two arbitrary sequences *a* and *b* of length *n* chips belonging to a code family *X* of size *K*, there are two meaningful ways to define the mean square cross-correlation value:

- By averaging over all relative phases for a single pair of sequences chosen from the family:

$$\overline{\rho_{a,b}^2(\tau)} = \frac{1}{n}\sum_{\tau=0}^{n-1}\rho_{a,b}^2(\tau)$$

- By averaging over all relative phases for each possible pair of sequences chosen from the family:

$$\overline{\rho_X^2(\tau)} = \frac{1}{nK(K-1)}\sum_{a\in X}\sum_{\substack{b\in X\\ b\neq a}}\sum_{\tau=0}^{n-1}\rho_{a,b}^2(\tau) = \frac{1}{K(K-1)}\sum_{a\in X}\sum_{\substack{b\in X\\ b\neq a}}\overline{\rho_{a,b}^2(\tau)}$$

Note that we have employed $\rho(\tau)$ as a generic (i.e. even or odd) cross-correlation function, although it can be argued that more general definitions apply in the case of the mean-square odd cross-correlation value because we have seen that in this case $\rho(\tau)$ depends upon the precise window of *n* contiguous chips upon which the averaging is performed.

Considering the former definition, it can be shown [66,67] that the mean square *even* cross-correlation is related to the *even* autocorrelation functions of the two individual sequences (assuming these to be binary sequences with values belonging to {+1,-1}) by the expression:

$$\overline{\theta_{a,b}^2(\tau)} = \frac{1}{n}\sum_{\tau=0}^{n-1}\theta_{a,b}^2(\tau) = \frac{1}{n}\sum_{\tau=0}^{n-1}\theta_{a,a}(\tau)\theta_{b,b}(\tau)$$

This expression can be used to set up lower and upper bounds on the value of the mean square *even* cross-correlation as illustrated in [66,67].

It should be clear that the mean square cross-correlation value corresponding to the former definition will depend upon the particular pair of sequences chosen for at least some families of codes (notably purely random sequences). However, this is not always the case. For instance, for any arbitrary *m*-sequences *a* and *b*, $\theta_{a,a}(0) = \theta_{b,b}(0) = n$, and $\theta_{a,a}(\tau) = \theta_{b,b}(\tau) = -1$, for $1 \le \tau \le n-1$. Thus, with reference to the previous equation, the value of

$$\overline{\theta_{a,b}^2(\tau)}$$

is the same irrespective of which two *m*-sequences of length *n* chips are chosen, and we have the result that:

$$\overline{\theta_{a,b}^2(\tau)} = \frac{n^2+n-1}{n} = \overline{\theta_X^2(\tau)}\,.$$

Clearly then, as far as *m*-sequences are concerned, the two definitions discussed above always yield identical results for the mean-square even cross-correlation value.

However, in order to compare the mean square cross-correlation of Gold and random codes, we will employ the latter definition and average over all sequences in the code family.

6.5.4.2.2 A comparison between Gold and random codes

The set of Gold codes of order *m* have length $n=2^m-1$ chips, [67], and, for any two such codes, there exists a three valued periodic cross-correlation function $\theta(\tau)$.

For *m* odd, it can be shown that:

$$\theta(\tau) = \begin{cases} 2^{(m+1)/2}-1 \\ -1 \\ -2^{(m+1)/2}-1 \end{cases} \quad \text{for} \quad \begin{cases} 25\% \\ 50\% \\ 25\% \end{cases} \quad \text{of all code phases } \tau.$$

Thus, the mean square periodic cross-correlation coefficient value is given by:

$$\overline{\theta_X^2(\tau)} = 0.25(2^{(m+1)/2}-1)^2 + 0.50 + 0.25(-2^{(m+1)/2}-1)^2$$
$$= 2^m + 1 = n + 2$$

For *m* even and $m \neq 0$ mod 4, it can be shown that:

$$\theta(\tau) = \begin{cases} 2^{(m+2)/2}-1 \\ -1 \\ -2^{(m+2)/2}-1 \end{cases} \quad \text{for} \quad \begin{cases} 12.5\% \\ 75\% \\ 12.5\% \end{cases} \quad \text{of all code phases } \tau$$

Thus, the mean square periodic cross-correlation coefficient value is given by:

$$\overline{\theta_X^2(\tau)} = 0.125(2^{(m+2)/2}-1)^2 + 0.750 + 0.125(-2^{(m+2)/2}-1)^2$$
$$= 2^m + 1 = n + 2$$

which is the same as for *m* odd, and is in fact equal to the number *K* of Gold Codes of length $n = 2^m-1$ chips.

Now, the mean square periodic cross-correlation coefficient value for a pair of purely random sequences of length *n* chips is simply the variance of the generic cross-correlation function $\rho(\tau)$ (since $E\{\rho(\tau)\}=0$) and is given by:

$$\overline{\theta_X^2(\tau)} = n.$$

Hence, the mean square periodic cross-correlation coefficient value for a pair of Gold sequences of length *n* (*n* large) is the same as that for a pair of purely random sequences of the same length. With reference to a previous equation, it is also apparent that, for large *n*, this value also applies to *m*-sequences.

It is important to stress that this result has been obtained by effectively averaging *over all relative phases for all possible pairs of sequences* of length $n = 2^m-1$ chips in a given family.

6.5.4.2.3 Other results

In [69], a different approach was adopted to that outlined in subsection 6.5.4.2.2. Mean square cross-correlation values (both odd and even) were obtained by *averaging over all relative phases for a single randomly chosen pair of sequences* of a given length from each family. Gold, Kasami (both small and large set) and *m*-sequences were employed in this investigation. The following conclusions were drawn:

- There is very little variation between calculated mean square even and odd cross-correlation values for a given code family in spite of the fact that the maximum absolute odd cross-correlation value is usually significantly larger than the maximum absolute even cross-correlation value.

- There is very little variation between calculated mean square cross-correlation values (whether even or odd) pertaining to different code families in spite of the fact that the maximum absolute cross-correlation values for the different families differ considerably.

These conclusions support and extend the observations of the preceding discussion. Furthermore, the observed relative invariance of mean square cross-correlation to the code family chosen appears to extend to families of non-linear codes.

In [70], mean square aperiodic cross-correlation parameters are investigated. An interesting result is that while the mean square aperiodic cross-correlation values for Gold and random codes are identical, those for m-sequences are found to be considerably better.

6.5.4.3 Comparisons based upon the probability of exceeding an absolute correlation threshold

6.5.4.3.1 Introduction

We have seen in subsection 6.5.4.2 that the mean square even (periodic) cross-correlation for a code length n is approximately equal to n for any arbitrary code family (or at least a set of codes with each member possessing a desirable (expected) autocorrelation function) if all codes pertaining to the family are considered in the averaging process. This invariance between code families is also apparent in the mean square odd cross-correlation, although, as a rule-of-thumb, the mean square odd cross-correlation will be slightly larger then the mean square even cross-correlation. These results can be taken to imply that the average performance of an asynchronous CDMA system is almost independent of the code family chosen.

However, such a conclusion can be misleading because, in practice, it is not the precise magnitude of the cross-correlation function which is important, but whether this magnitude exceeds a predefined threshold level. To clarify this argument, the probability of error P_e for individual bits on any individual traffic channel in an asynchronous CDMA system which is interference (rather than noise) limited can be written conceptually as:

$$P_e = P(|\theta(\tau)| > x_e(A))$$

in which:

- A represents the instantaneous number of active users in the system.
- x_e is some strictly monotonically decreasing positive function of A.

In general, the form of the function x_e will depend upon the code family employed. However. a very simple and pessimistic analysis can be performed which is not specific to any particular code family. Specifically, if there are A active users, then there are (A-1) interferers to any individual traffic channel. If each of the (A-1) interfering transmissions correlates with the wanted transmission to the same extent and in the same sense, we have that:

$$x_e = \frac{n}{A-1}$$

since the in-phase autocorrelation has magnitude n. Hence, the previous equation reduces to:

$$P_e = P(|\theta(\tau)| > n/(A-1)).$$

6.5.4.3.2 A Comparison Between Gold and random Codes

Figure 1 compares probability distribution functions for the even (periodic) cross-correlation functions of Gold and purely random codes.

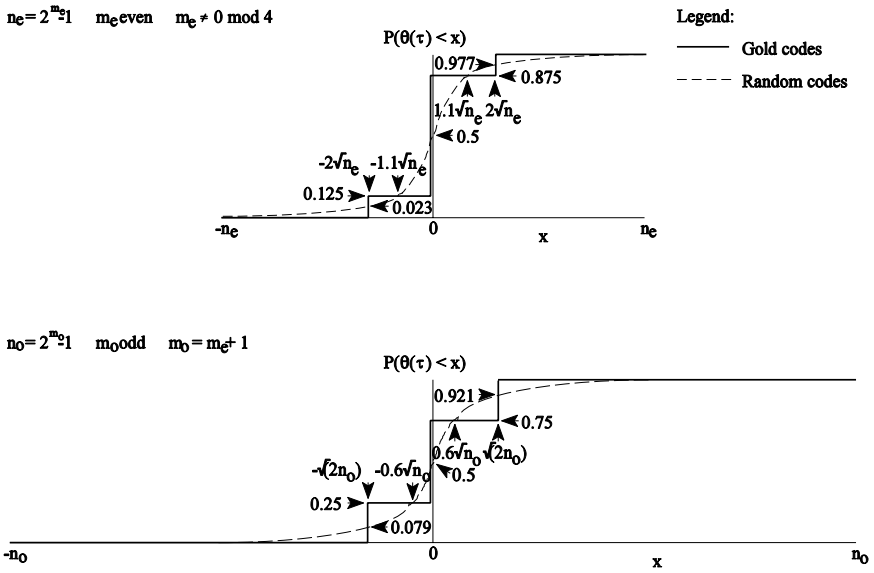

Figure : Probability Distribution Functions for the Even Cross-Correlation Functions of Gold and Purely Random Codes

While purely random codes can have any arbitrary length, Gold codes are constrained to have a length n of the form $n = 2^m$-1 and hence the comparisons illustrated in Figure 1 must also be constrained to the same values of n. Two distinct comparisons are made:

- One for $m = m_e$ even with m_e such that $m_e \neq 0 \bmod 4$.
- One for $m = m_0$ odd with m_0 such that $m_0 = m_e + 1$.

In these comparisons, we assume that n is relatively large. Then the exact even cross-correlation levels for Gold codes can be expressed approximately in terms of n as illustrated in Figure 1. Note that the three values which are possible for the even cross-correlation of Gold codes are exactly the same in both comparisons.

We saw in subsection 6.5.3 that the even cross-correlation function for purely random codes is, for n large, distributed according to:

$$\theta(\tau) \sim N(0,n).$$

Hence, the probability distribution function for the even cross-correlation function of purely random codes is given by:

$$P(\theta(\tau) < x) = \Phi(\tfrac{x}{\sqrt{n}})$$

in which Φ is the probability distribution function of a random variable distributed according to the standard normal distribution. Thus, the form of the probability distribution functions for purely random codes are as illustrated in Figure 1.

From Figure 1, it is clear that for:

- $1.1\sqrt{n} < x < 2\sqrt{n}$ in the case of m even and $m \neq 0 \bmod 4$,
- $0.6\sqrt{n} < x < \sqrt{(2n)}$ in the case of m odd,

purely random codes perform better than Gold codes in that the quantity $P(|\theta(\tau)|>x)$ is lower for purely random codes.

With respect to the discussion of subsection 6.5.4.3.1, this implies that purely random codes will perform better on average than Gold codes for intermediate loading of an asynchronous CDMA system, but worse for relatively low or high loading. However, in the case of relatively low loading, the system will probably be noise rather than interference limited so Gold codes are only likely to perform better than purely random codes in the case of relatively high loading. These conclusions are likely to be reinforced when we consider odd in addition to even cross-correlation functions as the odd cross-correlation functions of Gold codes are significantly inferior to the corresponding even cross-correlation functions; for purely random codes, the two types of correlation function are distributed identically as discussed in Section 6.5.3.

7. Key management for end-to-end security

7.1 Introduction

In this section we consider techniques for distributing keys for end-to-end security features.

In subsection 7.2 we propose a novel solution to the problem of providing trusted third party based key management for end-to-end encryption, in a way that meets legal requirements for warranted interception.

The ideas in subsection 7.2 are developed further in subsection 7.3, where we consider the use of multiple trusted third parties to achieve the same objectives as the protocol of 7.2, but where users do not need to trust any individual TTP to behave in a completely trustworthy way.

Subsection 7.4 gives a survey of research in the area of secret key agreement using public information. This is a topic on which several important research papers have appeared in the last two or three years, and, although the ideas appear to be difficult to implement, they could have profound effects on the design of mobile telecommunications systems security if they can be made practicable.

Finally, in subsection 7.5, we consider a variation on the theme of subsection 7.4, this time based upon the complexity of storing large amounts of information.

7.2 Trusted third party based key management

The text in this section is based on three papers, [71,72,73], prepared for publication as part of the 3GS3 project.

In this subsection we propose a novel solution to the problem of providing trusted third party services, such as the management of cryptographic keys for end-to-end encryption, in a way that meets legal requirements for warranted interception. Also included are a discussion of what might constitute a reasonable set of requirements for international provision of such services, an analysis of the cryptographic properties of the scheme, consideration of how it might operate in practice, and a generalisation of the scheme to provide for 'split escrow', .i.e. allowing a user to distribute trust over several TTPs.

7.2.1 Introduction

There has been much recent discussion on the question of how to meet users' requirements for security services, such as confidentiality and authentication, whilst at the same time meeting legitimate requirements of government agencies for access to communications; a survey of recent work can be found in an article by Denning and Branstad, [74]. The discussion has been largely prompted by the US government's Clipper proposals [75], as well as the increasing use of electronic means for transferring commercially sensitive data. On the one hand, users want the ability to communicate securely with other users, wherever they may be, and on the other hand, governments have requirements to intercept traffic in order to combat crime and protect national security. Clearly, for any scheme to be acceptable on a wide basis, it must provide the service that users want, as well as meeting the legal requirements in the territories it serves.

To create a platform that can be used to provide user services, it is anticipated that solutions will be based on the use of trusted third parties (TTPs) from which users can obtain the necessary cryptographic keys with which to encrypt their data or make use of other security services. Law enforcement agencies' requirements will be focused on the need to obtain the relevant keys from a TTP within their jurisdiction, so that they can decrypt precisely those communications that they are authorised to intercept.

In this subsection we propose a novel mechanism that will enable TTPs to perform the dual rôle of providing users with key management services and providing law enforcement agencies with warranted access to a particular user's communications. Unlike other proposals, the mechanism allows users to update their keys according to their own internal security policies. Moreover, it provides a framework for Diffie-Hellman key establishment which obviates the need for directories.

After briefly considering possible attacks on the mechanism, we list typical application requirements for such a scheme, and consider how well the proposed mechanism meets these requirements. It is important to note that the scheme described here has been designed to establish keys for providing end-to-end confidentiality services, and not for integrity, origin authentication or non-repudiation services; the appropriateness of the mechanism for providing these services is a matter for further study. We conclude by considering possible variants of the basic method, including a scheme allowing 'split escrow', and also how other proposed schemes for using TTPs in this way relate to the described method.

7.2.2 The Mechanism

The proposed mechanism is based upon the Diffie-Hellman algorithm for key exchange [98]. In order to simplify our description, we consider the mechanism only in relation to one-way communication (such as e-mail). The adaptation of the scheme for two-way communication is very straightforward.

More specifically we present the mechanism in the context of a pair of users *A* and *B*, where *A* wishes to send *B* a confidential message and needs to be provided with a session key to protect it. We suppose that *A* and *B* have associated TTPs *TA* and *TB* respectively, where *TA* and *TB* are distinct. Note that, since this scheme is intended to provide warranted access to user communications via the TTPs, we assume that each TTP is located within the jurisdiction of some intercepting authority, and

that each TTP operates subject to the regulations of that authority (typically one might expect such TTPs to operate within the terms of some kind of licence).

7.2.2.1 Initial requirements

Prior to use of the mechanism, *TA* and *TB* need to agree a number of parameters, and exchange certain information.

- Every pair of TTPs whose users wish to communicate securely must agree between them values *g* and *p*. These values may be different for each pair of communicating TTPs, and must have the usual properties required for operation of the Diffie-Hellman key exchange mechanism, namely that *g* must be a primitive element modulo *p*, where *p* is a large integer (satisfying certain properties). These values will need to be passed to any client users of *TA* and *TB* who wish to communicate securely with a client of the other TTP.

- Every pair of TTPs whose users wish to communicate securely must agree on the use of a digital signature algorithm. They must also each choose their own signature key/verification key pair, and exchange verification keys in a reliable way. Any user *B* wishing to receive a message from a user *A*, with associated TTP *TA*, must be equipped with a trusted copy of *TA*'s verification key (typically this would be provided by their own TTP *TB*, perhaps by means of a signed certificate).

- Every pair of TTPs whose users wish to communicate securely must agree a secret key $K(TA,TB)$ and a Diffie-Hellman key generating function *f*. This function *f* shall take as input the shared secret key and the name of any user, and generate for that user a private integer *b* satisfying $1 < b < p-1$ (which will be a 'private receive key' assigned to that user–see immediately below). The secret key $K(TA,TB)$ might itself be generated by a higher-level Diffie-Hellman exchange between the TTPs, or by any other bilaterally agreed method.

Given that *B* is to be provided with the means to receive a secure message from *A*, prior to use of the mechanism *A* and *B* need to be provided with certain cryptographic parameters by their respective TTPs.

- Using the function *f*, the secret key $K(TA,TB)$ and the name of *B*, both *TA* and *TB* generate the private integer *b* satisfying $1 < b < p-1$ (as described above). This key is known as *B*'s *private receive key*. The corresponding *public receive key* for *B* is set equal to $g^b \bmod p$. The private receive key *b* for *B* needs to be securely transferred from *TB* to *B* (like the other transfers discussed here, this can be performed 'off-line'). Note that *B* will be able to derive its public receive key from *b* simply by computing $g^b \bmod p$. Note also that this key can be used by *B* to receive secure messages from any user associated with *TA*; however, a different key pair will need to be generated if secure messages need to be received from users associated with another TTP.

- *A* must be equipped with a *send key pair*, for use when sending confidential messages to users associated with TTP *TB* (in fact this key pair could be used with many, perhaps all, other TTPs, as long as they share the values *g* and *p*). *A*'s TTP randomly generates a *private send key* for *A*, denoted *a* (where $1 < a < p-1$). *A*'s *public send key* is then set equal to $g^a \bmod p$. *TA* then signs a copy of *A*'s public send key concatenated with the name of *A* using its private signature key; this yields a certificate for *A*'s public send key. The signed certificate is then passed to *A*, together with a copy of *A*'s private send key *a* (this must be done using a secure channel between *A* and the TTP).

  In fact, in principle at least, *A* could generate the private send key *a* him/herself, and then only pass its *public* send key to *TA* (by some reliable means which does not need to preserve secrecy). *TA* would then sign a copy of *A*'s public send key concatenated with the name of *A* to yield a certificate for *A*'s public send key, which would then be passed back to *A*. The key escrow system would still work even though *A*'s TTP might not know *A*'s private send key. However, as we discuss in more detail below, the key escrow system works in a more flexible way if *A*'s TTP has access to *A*'s private send key, while giving the TTP the private send key of *A* does not give the TTP access to any more encrypted messages than if the TTP did not have access to this key.

- *A* must also be equipped with a copy of *B*'s public receive key. *B*'s private receive key *b* can be computed by *TA* using *f*, the name of *B*, and the key *K*(*TA*,*TB*). *TA* can then compute *B*'s public receive key as g^b, which can then be transferred in a reliable way from *TA* to *A*.

7.2.2.2 The mechanism itself

As we have seen, prior to use of the mechanism, *A* possesses the following information:

- *A*'s own private send key *a*;
- a certificate for *A*'s own public send key (g^a mod *p*), signed by *A*'s TTP *TA*;
- the public receive key (g^b mod *p*) for user *B*, and;
- the parameters *g* and *p*.

This information can be employed to generate a shared key g^{ab} mod *p* for protecting the confidentiality of a message to be sent from *A* to *B*. This key can be used as a session key, or, even better, as a key-encryption key (KEK). The KEK would then be used to encrypt a suitable session key. This latter approach has a number of advantages. For example:

- it would facilitate the sending of email to multiple recipients, since the message can be encrypted once under a random session key, and this session key can then be distributed to each recipient by encrypting it using the KEK, and
- it allows the use of a new key for each message.

User *A* then sends the following information to user *B*:

- the message encrypted using the session key (either g^{ab} mod *p* or a key encrypted using g^{ab} mod *p*),
- *A*'s public send key (g^a mod *p*) signed by *TA*, and
- the public receive key (g^b mod *p*) for user *B*

Once received, the public receive key g^b mod *p* allows user B to find its corresponding private receive key *b* (there will be a different receive key for each TTP with whose users *B* communicates). User *B* can then generate the (secret) session key g^{ab} mod *p* by raising *A*'s public receive key (g^a mod *p*) to the power of *B*'s own private receive key *b*, and thus can decrypt the received message.

A diagrammatic representation of the scheme is given in Figure 2.

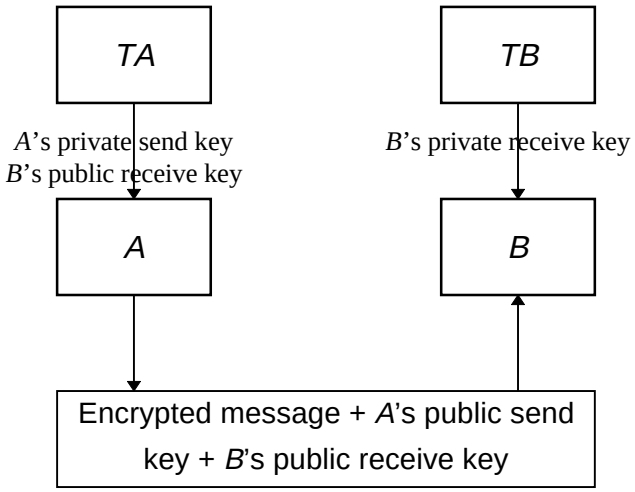

Figure 2: Use of the TTP scheme for one-way encrypted communication

7.2.2.3 Warranted interception

Should there be a warrant for legal interception of this communication, an intercepting authority can retrieve either the private send key of the 'sending user' or the private receive key of the 'receiving user' from the trusted third party within its jurisdiction, and use this in conjunction with the public receive key of the 'receiving user' or the public send key of the 'sending user', respectively, to find the session key for the encryption. There is no requirement for the intercepting authority to deal with any TTPs outside of its jurisdiction, or for any TTPs outside of its jurisdiction to know what is going on.

More specifically, suppose user *A* (served by TTP *TA*) has sent a message to user *B* (served by TTP *TB*). There are two cases to consider, namely depending on whether *TA* or *TB* is required to provide access to the encrypted message.

First suppose *TA* is required to provide access to this message. There are two ways in which *TA* could recover the shared key g^{ab} mod *p*, namely it can combine either:

- *B*'s private receive key *b* (generated from *K*(*TA*,*TB*) and the name of *B* using the key generating function *f*), with
- *A*'s public send key (g^a mod *p*), sent with the message,

or

- *A*'s private send key *a*, with
- *B*'s public receive key (g^a mod *p*), sent with the message.

Second suppose that *TB* is required to provide access to this message. Then, because *TB* will not have access to *A*'s private send key, there is only one way in which *TB* could recover the shared key g^{ab} mod *p*, namely by combining

- *B*'s private receive key *b* (generated from *K*(*TA*,*TB*) and the name of *B* using the key generating function *f*), and
- *A*'s public send key (g^a mod *p*), sent with the message.

When presented with the appropriate authorising information (e.g. a warrant), there are then two possible ways for the TTP to use this information to provide warranted access to communications. The TTP could pass the appropriate keys to the intercepting authority, and then take no further part in the interception process, or the TTP could use its escrowed key(s) to decipher messages presented to it by the intercepting authority, without revealing the keys themselves.

In order to assess the relative merits of these different approaches to providing warranted interception, we first need to consider four possible situations (where the first two correspond to what seem to be the most likely scenarios). We use notation corresponding to our discussion immediately above.

1. TTP *TA* is warranted to provide access to all outgoing communications from a user *A* for which it acts.
2. TTP *TB* is warranted to provide access to all incoming communications to a user *B* for which it acts,
3. TTP *TA* is warranted to provide access to all incoming communications (from users for which it acts) to a user *B* for which it does **not** act,
4. TTP *TB* is warranted to provide access to all outgoing communications (to users for which it acts) from a user *A* for which it does **not** act,

We first suppose that the TTP is required to provide keys to the intercepting authority. In case (1) it is sufficient for *TA* to provide the private send key(s) for *A*, and divulging these keys to the intercepting authority will not reveal information about any traffic not being sent by the user covered by the warrant. Note that, if *TA* did not possess the private send key for *A*, then it would be much more difficult for *TA* to provide warranted access to all *A*'s messages (it would be necessary for TTP to supply the session key for each individual recipient). In case (2), *TB* can supply *B*'s private receive

keys for each of the other relevant TTPs; as before, divulging these keys to the intercepting authority will not reveal any information about any traffic not being sent to *B*. In case (3), *TA* can supply *B*'s private receive key (which it can work out), again without revealing any information not covered by the warrant. Case (4) is the most problematic, since *TB* will not have access to *A*'s private send key. In this case (which is rather less likely than cases (1) or (2)), the only thing that *TB* can do is provide the intercepting authority with the key g^{ab} mod p for every other user *B* which *A* sends messages to. Thus in all but one, relatively unlikely, case, the TTP can very easily provide exactly the key which will enable the intercepting authority to gain access to the identified user's communications, without providing access to any communications which the intercepting authority is not entitled to.

The second approach to providing warranted access, i.e. having the TTP decipher messages 'on demand', avoids any of the problems we have just discussed. However, the main disadvantage of this approach is the increased amount of communication required between the TTP and the intercepting authority, and the potential delay in accessing enciphered information.

In the final analysis, the exact way in which the TTPs provide warranted access to communications is a political matter, and may even vary from domain to domain. The purpose of the above discussion is to show what options are available, and consider their technical advantages and disadvantages.

7.2.2.4 Properties of the mechanism

We next observe a few significant properties of the proposed mechanism.

- First note that a user can arrange for his/her *send key pair* to be changed at any time. A user simply requests his/her TTP to generate a new key pair for him/herself, which is then passed by the TTP to the user (along with a signed certificate).
- No directories are required to make the system work. An entity wishing to send a message only needs to obtain the public receive key for the intended recipient from his/her own TTP, who can generate this information merely from the name of the recipient and the identity of the recipient's TTP. A recipient of an enciphered message will, given the information contained in the message, possess all the data necessary to obtain the session key, without further reference to any third parties.
- Whilst, given the description above, receive key pairs are apparently fixed, by including the year (or month and year) within the scope of the key generating function *f*, all receive key pairs can automatically be updated at regular intervals. We discuss an option of this type in more detail in subsection 7.2.4.1 below.

7.2.2.5 Possible methods of attack

We conclude this discussion of the mechanism by considering what approaches might be used to attack the scheme. First observe that the scheme is based on the Diffie-Hellman key exchange scheme, which has withstood detailed scrutiny over a period of time. The only means of attack on Diffie-Hellman of relevance here would appear to be the Burmester attack, [76], which we now discuss.

The basic idea of the Burmester attack (in the context of the scheme presented here) is as follows. Suppose user *A* has public send key g^a, and user *B* has public receive key g^b mod *p*. Then the key to be used to protect messages sent from *A* to *B* will be g^{ab} mod *p*. Now suppose that a third user, *C* say, manages to persuade *B* that its public send key is g^a mod *p* (we consider below ways in which this might occur). This means that the session key used for encrypting messages sent from *C* to *B* is g^{ab} mod *p*. *C* next claims to have temporarily lost its copy of the session key, and asks *B* to supply a replacement copy (by some secure means). *B*, believing that *C* is entitled to a copy of this key complies, and now *C* has the means to decipher all the traffic sent from *A* to *B*.

As discussed in Burmester's paper, [76], there are many ways in which such an attack can be avoided, even without considering how *C* manages to persuade *B* that g^a mod *p* is really *C*'s public send key (we return to this latter point in a moment). First and foremost, *B* should never divulge the session key g^{ab} mod *p*, even if *B* believes that *C* is really entitled to it (since if *C* is really entitled to it, then why cannot *C* recompute it for him/herself?). Second, in a practical implementation users should be

prevented from directly accessing session keys, thereby preventing them from divulging secrets accidentally or deliberately. This is probably a good idea in most practical security systems.

We now return to the key part of the above discussion, namely the means by which *C* manages to persuade *B* that *A*'s key is really *C*'s key (even though *C* does not know the private key *a*). There would appear to be three main ways in which this might occur. We consider each of them, and show that in each case either simple measures can prevent such attacks, or that the attack does not apply.

1. The first possibility is that *C* persuades *A*'s TTP (*TA*) that *C*'s public send key is g^a mod *p*, and gets *TA* to sign a certificate to this effect. To avoid this, a TTP must always get proof that a user possesses the private key corresponding to a public key before signing a certificate to the effect that this public key belongs to the user. This could be done be asking the user to utilise the private key to sign some data which could then be checked using the public key. This is in any case already accepted as good practice for the operation of a certification service.

2. The second possibility is that *A*'s TTP (*TA*) colludes with *C* and generates a certificate to the effect that *C*'s public send key is g^a mod *p*. Whilst this may be possible, *TA* already has the means to read *A*'s messages, and hence gains nothing by such an attack! The whole point of any key management system based on TTPs is that a user must trust the TTP they appoint to act on their behalf. We are thus entitled to ignore this case.

3. The third possibility is that *C* persuades some other TTP (*TC* say) that *C*'s public send key is g^a mod *p*, and gets *TC* to sign a certificate to this effect. However the above attack will no longer work in this case, since *B*'s private receive keys for users *C* and *A* will be different, since they are served by different TTPs. This should be clear by observing that every user's private receive key is computed as a function of a key shared by their respective TTPs. Hence, although *A* and *C* apparently share a *send key*, the session key for protecting messages sent from *A* to *B* will be different from the session key used for protecting messages sent from *C* to *B*, and hence *B*, however co-operative he/she might be, cannot reveal any useful information to *C*.

The above analysis shows that given that, before generating a certificate, a TTP always checks that a user possesses the private key corresponding to the public key which they claim as their own, then the Burmester attack does not apply.

7.2.3 A Typical Set of Requirements on a Trusted Third Party Scheme

Clearly, the definition and agreement of a set of requirements acceptable across a broad set of countries is largely a political process. However, we can give a set of typical or likely requirements on which to base an analysis of the suitability of the proposed mechanism.

Use of the scheme should provide visible benefits for the user. The design and operation of the scheme means that the TTPs are capable of offering their services to users on a commercial basis. By signing up to a licensed TTP, the user will be able to communicate securely with every user of every TTP with whom his TTP has an agreement. The user would potentially be able to choose from a number of TTPs in his home country, thus increasing his trust in the TTP.

The scheme should allow national and international operation. The proposed scheme achieves this by ensuring that the intercepting authority can obtain the required keys from a TTP within its jurisdiction.

Details of the scheme should be public. This is achieved for the proposed scheme by the publication of papers [71,72,73].

The scheme should be based on well known techniques, and Diffie-Hellman certainly qualifies.

All forms of electronic communication should be supported. The proposed scheme can easily be adapted to include two-way communication such as voice telephony.

The scheme should be compatible with laws and regulations on interception, as well as on the use, export and sale of cryptographic mechanisms. This matter is the subject of further study, but no problems have yet been identified.

Access must be provided to the subject's incoming and outgoing communication, where a warrant is held. This is clearly achieved for the proposed scheme, as the subject's TTP will be able to provide the appropriate session keys.

The scheme should support a variety of encryption algorithms, in hardware and software. As the proposed scheme deals solely with key management, any suitable encryption algorithm can be used, as long as it is available to the recipient and legitimate interceptors. The best way to achieve this may be to use a standard list of algorithms, such as the ISO register.

An entity with a warrant should not be able to fabricate false evidence. This is particularly applicable in countries where intercepted communications are admissible as evidence in court. The proposed scheme as it stands does not meet this requirement, but the provision of digital signatures as an additional service by the TTP will allow it to be met.

Where possible, users should be able to update keys according to their own internal policies. The proposed scheme allows a user to have new send key pairs generated as often as wished. The receive keys, which are generated deterministically based on the TTPs' shared key and the user's identity, are more permanent, and change only if the TTPs' shared key or the user's identity changes. However, as we have already noted, if there is a requirement for receive keys to be changed at regular intervals, a date stamp could be included within the scope of the key generating function *f*. This would have the advantage that any private receive key provided to an intercepting authority would have only a limited period of validity, meaning that the warranted interception capability could only last for a certain time period before needing to be renewed.

Abuse by either side should be detectable by the other. We believe that this is the case for the proposed scheme, although abuse by collusion between the two sides may still be possible. The main disincentive to such abuse may be the 'shrink-wrapped' provision of the software, which we would expect to be bundled in with, say, an email system or other telecommunications software.

Users should not have to communicate with TTPs other than their own. The only communication required in the proposed scheme is with the user's own TTP.

On-line communication between TTPs should not be required. The independent generation of the receive keys in the proposed scheme means that no such communication is required for the proposed scheme.

7.2.4 Two variations on the basic mechanism

The proposed scheme can almost certainly be modified in many ways. We briefly present two such modifications, and consider their associated advantages and disadvantages.

7.2.4.1 Time bounding of TTP keys

As we have already discussed, in the scheme described above a user's receive key pair is apparently fixed, since it is generated as a deterministic function of a secret key shared by two TTPs and the name of the receiving user. We now describe one way in which this problem can be overcome though the use of date-stamps.

- The modified system requires the use of two key-generating functions *f* and *g* instead of one (although *f* and *g* might be the same function).
- Whenever user *A* requests his/her TTP *TA* to supply a copy of the public receive key of entity *B*, *TA* first computes *B*'s *permanent* private receive key *b* using *f* with inputs:

    1. the identity of *B*,
    2. the secret key shared by *TA* and *TB*,

    just as before. *TA* then uses *g* with the two inputs *b* and a current date-stamp to generate *today*'s private receive key for *B*, which we denote b'. *TA* then passes *B*'s public receive key for today ($g^{b'}$ mod *p*) to *A*.
- *A* uses *B*'s public receive key for today (in conjunction with *A*'s private send key) to compute a session key $g^{ab'}$ which is used to encrypt the message to *B*. *A* also sends a copy of today's public

receive key with the encrypted message, along with the current date-stamp. Ideally *TA* should bind the date-stamp to *B*'s public receive key for today using a digital signature.

- As before, *B* will have been equipped with its own permanent private receive key *b* by its TTP *TB*. *B* can then use the function *g* to compute b' from *b* and the current date, and hence compute $g^{ab'}$, and thus decrypt the message.

The main advantages of this modified scheme are as follows.

- Every user's receive key pair will automatically change every day.
- Time-bounding of warrants could be enforced by only providing receive key pairs for the days specified in the warrant to an intercepting authority.

The main disadvantage is as follows.

- The TTP will need to pass several key pairs to an intercepting authority to provide access to communications for a period of time exceeding one day.

7.2.4.2 A scheme allowing split escrow

In an environment where commercial TTPs will be looking to offer additional services to their users, it is possible that some users will want the extra reassurance offered by having their keys shared between a number of independent TTPs. The proposed protocol is easily adaptable to provide this feature. For instance, the ideas of Micali [77] for adding secret sharing on top of existing schemes could be adopted. We now propose another solution which has the advantage of reducing the number of key pairs that message originators need hold.

7.2.4.2.1 Operation of the modified mechanism

For this scheme to operate, and unlike Micali's scheme, [77], every user *A* (with identity ID_A) will have a distinct modulus p_A and base g_A, used to secure all messages originated by *A*. These are computed as deterministic functions of ID_A, e.g. $p_A = F_p(ID_A)$ and $g_A = F_g(ID_A)$, where F_p and F_g are universally agreed functions.

Now suppose that each user subscribes to a set of TTPs, which the user is prepared to trust collectively but perhaps not individually. Thus, as previously, in the context of the situation where user *A* wishes to send a secret message to user *B*, we suppose that *A* subscribes to the set of TTPs $\{X_i\}$, and that *B* subscribes to the set of TTPs $\{Y_j\}$. Each of the TTPs X_i will need to know the identities of all members of the sets $\{X_i\}$ and $\{Y_j\}$, and each pair of TTPs (X_i,Y_j) share a secret key which we denote by K_{ij}. Observe that each of the TTPs in the set $\{X_i\}$ will need to be within the jurisdiction of the intercepting authority within which *A* resides, and similarly for $\{Y_i\}$.

Before *A* can send a secret message to *B*, two key pairs will need to be established, as we now describe.

- ***A* will need to be equipped with a send key pair**. Each of the TTPs X_i generates a part of *A*'s private send key; denote the part generated by X_i as S_{Ai}. Each X_i also computes their part of *A*'s public send key as

$$P_{Ai} = g_A^{S_{Ai}}.$$

  Each TTP X_i now signs a concatenation of the name of *A* with their part of *A*'s public key, and passes the resulting certificate, denoted $<P_{Ai}>$, together with S_{Ai} to A. A can now compute his/her private send key as

$$S_A = \sum_i S_{Ai},$$

  and *A*'s public send key will be

$$P_A = g_A^{S_A} = \prod_i g_A^{S_{Ai}} .$$

- ***B* will need to be equipped with a receive key pair (and *A* will need to be given a copy of *B*'s public receive key)**. Each pair of TTPs (X_i,Y_j) now use a function *f*, taking as input their shared secret key K_{ij} and the identity of *B* (ID_B), to generate a part of *B*'s private receive key which we call S_{Bij}. I.e.

  $$S_{Bij} = f(K_{ij},ID_B).$$

  For each *j*, the values of S_{Bij} will be sent (securely) from TTP Y_j to *B*. *B*'s private receive key for use when receiving messages from clients of X_i will then be

  $$S_{Bi} = \sum_j S_{Bij} .$$

  Each of the TTPs X_i will perform the same calculation as *B*, and will also compute

  $$P_{Bi} = g_A^{S_{Bi}} .$$

  Each of these values will then be passed to *A* who can then calculate *B*'s public receive key as

  $$P_B = \prod_i P_{Bi} .$$

  It is important to note that the private receive key 'components' S_{Bi} are independent of the identity of *A*, but the public receive keys will vary depending on who the sender is.

The session key to be used to secure messages sent from A to B will then be

$$g_A^{S_A S_B} = (P_B)^{S_A} = (P_A)^{S_B}$$

and hence can be calculated by both *A* and *B*. The enciphered message will then be sent accompanied by the certified pieces of A's public send key <P_{Ai}> together with *B*'s public receive key P_B.

Key escrow will now operate in an exactly analogous way to that described in subsection 7.2.2.3 above, with the exception that each TTP within a domain will now be required to supply their parts of either the send private key or the receive private key.

7.2.4.2.2 Properties of the modified mechanism

This modified version of the mechanism has the following properties.

- This modified scheme has a possible advantage over the scheme described in subsection 7.2.2.2, even when each user only has one TTP. In this modified scheme users need only store one private receive key for each other TTP in the system, and only one private send key. The storage requirement for receive keys is thus unchanged, and the storage requirement for send keys is potentially reduced (depending on the number of different values of *g* and *p* used in the basic scheme).

- It is not possible for a user *A* to choose their own 'modulus' p_A and 'base' g_A, because *A* might include hidden structure within them which would allow access to other users' private receive keys. For example, p_A might be chosen so that some polynomial with small coefficients has a root modulo p_A, thereby facilitating the use of the Number Field Sieve. In addition, p_A must be prime, as its deterministic construction might otherwise compromise its factorisation.

- A user should probably not be permitted to change his/her modulus p_A, since that would potentially provide access to several different public keys computed from the same private key using different moduli. This might, at least in principle, lead to an attack on the private key using the Chinese Remainder Theorem.

- A possible alternative to deterministic generation of moduli is for a TTP to generate (and sign) a modulus p_A for *A*'s use. This would then need to be passed to each of the TTPs X_i, and also with each message sent by *A*. Such a scheme has the risk that it might be possible for a user to obtain several different moduli, and hence deterministic generation is probably preferable.
- *A* need not use the same set of TTPs all the time. For example, given that contacting a long list of TTPs may be expensive and/or inconvenient, the sender may choose to use the full set of TTPs only for particularly sensitive messages.
- It would be preferable to allow users to include a single public key with their message, signed by all of their TTPs, rather than the individual components specified above; apart from anything else this would reduce the communications overhead. However, it is not clear whether this can be done without increasing the possibility of 'cheating' the escrow system; it would also probably require the TTPs to communicate amongst themselves, which is not necessary with the current scheme.

7.2.5 Options and Other Issues

7.2.5.1 Trusting TTPs

The receiving party must trust the sending party's TTP, in order to verify the sending party's public key, and also because the sender's TTP can generate the receiver's private key. However, this trust only concerns communications between the receiver and senders belonging to that TTP.

There may also be a need for a certification hierarchy to identify a common point of trust for different TTPs; alternatively, all TTPs could manage their inter-relationships by bilateral agreement.

7.2.5.2 The Choice of Values

There has been considerable discussion in the literature on the benefits of using a composite modulus for Diffie-Hellman. This, and other matters such as the length of the modulus *p* and the primitive element *g*, are beyond the scope of the discussion here.

7.2.5.3 Commercial Value

The proposed scheme relies entirely on its perceived value to users in order to be taken up. Service providers will want to recover the cost of setting up the service from their customers. Therefore the scheme must be able to provide value-added end-to-end services that users want. Further investigation is required to assess the level of demand for services such as:

- end-to-end encryption;
- end-to-end authentication;
- non-repudiation of sent messages;
- message integrity.

Given that users will be paying for these services, they will expect a sufficient level of security. In the event of security failure with financial impact on the user, he will expect to be able to recover this, either via his insurers or from the organisation running the TTP. This makes running a TTP a potentially expensive business, unless the financial risks run by the TTP can be adequately protected against. If TTPs are not commercially viable, then the scheme will not be viable.

7.2.5.4 Combined two-way Session Key

The two-way version of the proposed scheme provides two keys for communication: one for each direction. These could be combined to form a single session key, or just one of the keys could be used. The advantages and disadvantages of this are a matter for further study.

7.2.6 Other Published Schemes

We conclude this discussion by briefly indicating how two other key establishment schemes relate to the scheme described above.

7.2.6.1 The Goss Scheme

A scheme designed by Goss has been patented in the US [78]. In this scheme, a shared secret key is established by combining two Diffie-Hellman exponentiations using fixed and varying (per session) parameters. At first sight, this appears to correspond to the receive and send keys in the proposed scheme. However, the Goss scheme uses a universal modulus and primitive element. If x and x' are A's fixed and variant keys, and y and y' are B's, then the shared key is calculated as

$$\alpha^{xy'} \oplus \alpha^{x'y}$$

This could be viewed as a variant of the proposed two-way protocol whereby a universal modulus and primitive element are used and the two keys are combined by XOR-ing them.

7.2.6.2 Yacobi Scheme

The scheme of Yacobi [79] is almost identical to the Goss one, but uses a composite modulus, and combines the session keys by modular multiplication rather than XOR-ing.

7.3 Key management using multiple TTPs

The text in this section is based on a paper, [80], prepared for publication as part of the 3GS3 project.

In this subsection we present a key escrow mechanisms which meet possible requirements for international key escrow, where different domains may not trust each other. In these mechanisms multiple third parties, who are trusted collectively but not individually, perform the dual role of providing users with key management services and providing authorised agencies in the relevant domains with warranted access to the users' communications. We propose two key escrow mechanisms, both designed for the case where the pair of communicating users are in different domains, in which the pair of users and all the third parties jointly generate a cryptographic key for end-to-end encryption. The fact that all entities are involved in the key generation process helps make it more difficult for deviant users to subvert the escrowed key by using a hidden 'shadow-key'. The first mechanism makes use of a single set of key escrow agencies moderately trusted by mutually mistrusting domains. The second mechanism uses a transferable and verifiable secret sharing scheme to transfer key shares between two groups of key escrow agencies, where one group is in each domain.

7.3.1 Introduction

7.3.1.1 Key escrow in mutually mistrusting domains

In modern secure telecommunications systems there are likely to be two contradictory requirements. On the one hand users want to communicate securely with other users, and on the other hand governments have requirements to intercept user traffic in order to combat crime and protect national security. A key escrow system is designed to meet the needs of both users and governments, where a cryptographic key for user communications is escrowed with a key escrow agency (or a set of agencies) and later delivered to government agencies when lawfully authorised. Following the US government's Clipper proposals, [75], a number of key escrow systems have recently been proposed, and for an overview of the field, the reader is referred to [74].

When users communicate internationally, there is a potential requirement to provide the law enforcement agencies of all the relevant countries, i.e. the originating and destination countries for the communication, with warranted access to the user traffic. For example, a global mobile telecommunications system might provide an end-to-end confidentiality service to two mobile users in two different countries, and law enforcement agencies in both these countries might independently wish to intercept these communications. To make matters more complicated, these two countries will typically not trust one other (such domains are referred to as mutually distrusting countries in [81]); for example, a law enforcement agency in one country might not wish to let their counterpart in any other country know that a particular user's communications are being intercepted.

We are concerned here with international key escrow, and we assume throughout that the countries involved do not trust one another; for the maximum generality we refer to *domains* instead of countries throughout. We also refer to *interception authorities* where we mean bodies such as law enforcement agencies who may be given the right to access communications within a single domain. Finally we refer to *escrow agencies* or *Trusted Third Parties (TTPs)* who will be responsible for maintaining all the information necessary to provide access to interception agencies, when presented with the appropriate legal authorisation.

We now state our requirements for key escrow in an international (i.e. a multi-domain) context.

1. No domain can individually control the generation of an escrowed key, and hence the escrowed key cannot be chosen by entities in only one domain and then transferred to the other domain.

2. The interception authorities in any domain can gain access to an escrowed key without communicating with any other domain, i.e. the key has to be capable of being escrowed in all relevant domains independently.

3. The entities in any domain can ensure the correctness and freshness of the escrowed key.

7.3.1.2 Prior approaches

In subsection 7.2 a key escrow mechanism suitable for international use, called the 'JMW' mechanism for short, was described. In that scheme every user has an associated TTP. If two users, communicating with each other securely by using end-to-end encryption, are located in different domains then the relevant pair of TTPs (one in each domain) collaboratively perform the dual role of providing the users with key management services and providing the two interception agencies with warranted access to the users' communications. A session key for end-to-end encryption is established based on Diffie-Hellman key exchange [98]. An asymmetric key agreement pair for one user (the receiver) is separately computed by both TTPs (one in each domain) using a combination of a secret key shared between them and the receiver's name, and another asymmetric key agreement pair for the other user (the sender) is generated by himself. The receiver computes the session key by combining his private key (transferred securely from his own TTP) with the sender's public key (sent with the encrypted message). The sender computes the same session key by combining his private key with the receiver's public key (obtained from the sender's own TTP). Interception agencies in each domain can retrieve the session key from the TTP in the same domain.

Note that this mechanism meets the three requirements for key escrow listed above. However, it requires the following assumptions about trust relationships among the users, TTPs and interception agencies.

1. Each user believes that their own TTP (as well as the TTPs of any other users with which they communicate) will issue proper key agreement values and certificates, and will not reveal the escrowed key illegally.

2. Each TTP believes that the user, as a sender, will provide the correct public key (matching the secret key they use for securing messages they send).

3. Each TTP believes that the other TTP contributes proper key agreement values and certificates, and will not reveal the escrowed key illegally.

4. Each interception agency believes that the TTP in its domain will provide the correct escrowed key when requested.

In [81], Frankel and Yung give a different scheme for international key escrow, which requires a key escrow agency (or agencies) to be trusted by more than one domain.

7.3.1.3 Another approach

In this subsection we suppose that, in some environments where international key escrow is required, TTPs may not be trusted individually to provide proper contributions to an escrowed key and to reveal the key legally, and users also may not be trusted to provide proper contributions to an escrowed key. Thus the scheme here meets some of the same possible requirements as the scheme described in subsection 4.4.

We consider two related key escrow systems with the following properties.

1. *The schemes use a set of moderately trusted third parties instead of a single TTP, in an effort to prevent a single TTP from corrupting an escrowed key.* For the purposes of our discussion here, moderately trusted third parties are trusted collectively, but not individually, by users, interception agencies and another set of TTPs.

   Key splitting schemes have previously been used for splitting an escrowed key into n shares escrowed by n agencies in proposed key escrow systems (e.g. see [74,82,83,84,85]); we also make use of a k out of n threshold scheme. Such a scheme allows any subset of k of the n escrow agencies to affect the recovery of a complete key, but prohibits any group of fewer than k agencies from recovering a complete key.

2. *They use a verifiable secret sharing scheme in order to prevent deviant users from subverting the secret sharing scheme by providing improper shares.* Such a scheme has previously been adopted in a key escrow system to let a group of key escrow agencies verify that they have valid shares [83].

3. *They use an affine expansible verifiable secret sharing scheme to let users and third parties jointly generate an escrowed key, thus preventing deviant users from obtaining a 'shadow-key' (not available to the escrow agency).*

4. *The second scheme makes use of a transferable verifiable secret sharing scheme to transfer shares between two sets of key escrow agencies which may not trust each other*.

The remainder of this part of TR2 is subdivided as follows. In subsection 7.3.2, we present a transferable verifiable secret sharing scheme and an affine expansible verifiable secret sharing scheme based on the Shamir secret sharing scheme, [35], and the Pedersen verifiable secret sharing scheme, [86]. We then propose two mechanisms for international key escrow in subsection 7.3.3. The first, which incorporates Frankel and Yung's idea, [81], makes use of a single group of key escrow agencies moderately trusted by mutually mistrusting domains. The second scheme, which is an alternative to the JMW mechanism, adopts the transferable and verifiable secret sharing scheme to transfer shares between two sets of moderately trusted key escrow agencies, one set within each of two mutually mistrusting domains. In both mechanisms, users and key escrow agencies jointly generate an escrowed key by using the affine expansible verifiable secret sharing scheme.

In subsection 7.3.4, we consider possible trust relationships among the three types of entity involved in an international key escrow system, namely moderately trusted third parties, potentially untrustworthy users and multiple mistrusting domains. We conclude in subsection 7.3.5.

7.3.2 Verifiable Secret Sharing

In this subsection we first briefly describe the Shamir secret sharing scheme [35] and the Pedersen verifiable secret sharing scheme [86]. We then discuss how to transfer a shared secret between two domains, and also how to share an affine function of a shared secret, using modifications of the Shamir and Pedersen schemes. This work will provide the basis for the key escrow schemes described subsequently.

7.3.2.1 The Shamir scheme

A (k,n)-threshold secret sharing scheme is a protocol in which a *dealer* distributes partial information (a share) about a secret to each of n participants such that:

- no group of fewer than k participants can obtain any information about the secret, and
- any group of at least k participants can compute the secret.

We now describe the Shamir (k,n)-threshold secret sharing scheme, [35]. Suppose p and q are large primes such that q divides p-1, and g is an element of order q in Z_p. It is assumed that p, q and g are publicly known. These parameters will be used throughout this part of TR2. Unless otherwise stated all arithmetic will be computed modulo p.

Let the secret s be an element of Z_q. In order to distribute s among P_1, ...,P_n (where $n < q$) the dealer chooses a polynomial of degree k-1:

$$f(x) = a_0 + a_1x + ... + a_{k-1}x^{k-1},$$

where $f \in Z_q[x]$ and $a_0 = s$. Each participant P_i ($1 \le i \le n$) receives $s_i = f(x_i)$ as his private share, where $x_i \in Z_q$-{0} is public information about P_i ($x_i \ne x_j$ if $i \ne j$).

Any k participants (without loss of generality we assume that they are P_1, P_2, ..., P_k) can find $f(x)$ by the interpolation formula,

$$f(x) = \sum_{i=1}^{k} \left(\prod_{h \ne i} \frac{x - x_h}{x_i - x_h}\right) f(x_i) = \sum_{i=1}^{k} \left(\prod_{h \ne i} \frac{x - x_h}{x_i - x_h}\right) s_i$$

Thus

$$s = f(0) = \sum_{i=1}^{k} (\prod_{h \neq i} \frac{x_h}{x_h - x_i}) s_i$$

7.3.2.2 The Pedersen scheme

Assume that a dealer has a secret $s \in Z_q$ and corresponding public value $h = g^s$. This secret can be distributed to and verified by P_1, ..., P_n, in the following way:

1. The dealer computes shares s_i using the Shamir secret sharing scheme by first choosing a polynomial $f(x) = a_0 + a_1 x + ... + a_{k-1} x^{k-1}$ over Z_q satisfying $a_0 = s$, and then computing $s_i = f(x_i)$ ($1 \leq i \leq n$). Here x_i is public information about P_i, as previously.

2. The dealer sends the share s_i secretly to P_i ($1 \leq i \leq n$) and broadcasts a verification sequence

   $$V = (g^{a_0}, g^{a_1}, \ldots, g^{a_{k-1}})$$

   to all n participants.

3. Each P_i ($1 \leq i \leq n$) computes

   $$h_i = \prod_{j=0}^{k-1} (g^{a_j})^{(x_i)^j},$$

   and verifies whether

   $$h_i = g^{s_i}.$$

   If this does not hold then P_i broadcasts s_i and stops. Otherwise P_i accepts the share.

4. Any k participants, who have accepted their shares, can find s as described in the Shamir secret sharing scheme above.

7.3.2.3 Transferable verifiable secret sharing

We now consider how to transfer a shared secret between two groups of participants. We start by stating our requirements for a (k,m,n)-transferable verifiable secret sharing scheme, where k, m, and n are positive integers satisfying $1 < k \leq \min\{m,n\}$.

- A secret s shared by m participants P_1, ..., P_m needs to be transferred to, and then shared by, another n participants Q_1, ..., Q_n.
- The participants Q_j ($1 \leq j \leq n$) must be able to verify their own private shares without communicating with other participants in the same domain.
- Any group of at least k participants in Q_1, ..., Q_n, who have accepted their shares, can compute s.
- No group of fewer than k participants in Q_1, ..., Q_n can obtain any information about s.

We now present a transferable verifiable secret sharing scheme based on the Shamir and Pedersen schemes.

Algorithm 7.3.1

Assume that m participants P_i ($1 \leq i \leq m$) share a secret $s \in Z_q$ using the Pedersen scheme. This secret can be transferred to and verified by another n participants Q_j ($1 \leq j \leq n$), in the following way:

1. Each P_i ($1 \leq i \leq m$) computes new shares s_{ij} ($1 \leq j \leq n$) using the Shamir secret sharing scheme by:

- first choosing a degree k-1 polynomial $f_i(x) = a_{i0} + a_{i1}x + ... + a_{i(k-1)}x^{k-1}$ over Z_q satisfying $a_{i0} = s_i$, and
- then computing $s_{ij} = f_i(x_j)$. Here x_j is public information about Q_j.

2. P_i $(1 \le i \le m)$ sends s_{ij} secretly to Q_j $(1 \le j \le n)$ and broadcasts a verification sequence

$$V_i = (g^{a_{i0}}, g^{a_{i1}}, ..., g^{a_{i(k-1)}})$$

to all n participants Q_1, ..., Q_n.

3. On receipt of s_{ij} and V_i $(1 \le i \le m)$, Q_j $(1 \le j \le n)$ computes

$$h_{ij} = \prod_{\ell=0}^{k-1} (g^{a_{i\ell}})^{(x_j)^\ell},$$

and verifies whether

$$h_{ij} = g^{s_{ij}}.$$

If this does not hold, Q_j broadcasts s_{ij} and stops. Otherwise Q_j accepts the share.

Theorem 7.3.2

The above algorithm has the following properties.

1. Any group of at least k participants in Q_1, ..., Q_n, who have accepted their shares following Algorithm 7.3.1, can find s_i $(1 \le i \le m)$, and hence compute s.
2. No group of fewer than k participants in Q_1, ..., Q_n can obtain any information about s_i $(1 \le i \le m)$ or s.
3. Each Q_j $(1 \le j \le n)$ can verify s_{ij} $(1 \le i \le m)$ and g^s without communicating with other participants in the same domain.

Proof

All three parts of the theorem hold by using precisely the same arguments as used to prove the same statements for the Pedersen scheme.

This scheme will be used to transfer a partial escrowed key from a set of TTPs in one domain to another set of TTPs in a second domain in Mechanism 7.4.7 described in the next subsection. The two groups of participants do not have to trust each other. If fewer than k participants in any domain follow the scheme, the secret transfer cannot be successful, but no one can subvert the algorithm by forcing anyone else to accept a fraudulent secret.

7.3.2.4 Affine expansible verifiable secret sharing

We now consider an *affine expansion* of threshold secret sharing. We start by stating our requirements for 'affine expansion'.

- A secret $s \in Z_q$ is shared by m participants P_1, ...,P_m. Its affine function $w = as+b$, where $a, b \in Z_q$ and $a \neq 0$, needs to be shared by the same participants. Here a and b are public information.
- No group of fewer than k participants can obtain any information about w.
- Any group of at least k participants can compute w.

We now present an affine expansible verifiable secret sharing scheme based on the Shamir and Pedersen schemes.

Algorithm 7.3.3

Assume that m participants P_i ($1 \le i \le m$) share a secret $s \in Z_q$ using the Pedersen scheme, and know public information $a \in Z_q$-{0} and $b \in Z_q$. A new secret $w = as+b \in Z_q$ can be shared and verified by the same m participants without communicating with one another. The new shares w_i are

$$w_i = as_i+b.$$

The corresponding public keys are

$$g^{w_i} = g^{as_i+b} = (g^{s_i})^a g^b, \text{ and}$$

$$g^{w} = g^{as+b} = (g^{s})^a g^b.$$

Theorem 7.3.4

The above algorithm has the following properties.

1. It meets the requirements for affine expansible secret sharing.
2. P_i ($1 \le i \le m$) can verify w_i ($1 \le i \le m$) and g^w without communicating with the other participants.

Proof

This theorem again follows using precisely the same arguments as are used to establish the properties of the Pedersen scheme.

This scheme will be used to let third parties provide a contribution to an escrowed key in Mechanisms 7.3.5 and 7.3.7 described below. Because the contribution is not known to users, it is difficult for the users to subvert the escrowed key by using a hidden 'shadow-public-key', the corresponding 'shadow-private-key' of which cannot be computed by using a real key pair and a 'shadow-public-key' [83].

7.3.3 Escrowed key agreement

7.3.3.1 Assumptions

We make the following assumptions for our model of an international key escrow system.

- Two entities *A* and *B*, located in mutually mistrusting domains, want to communicate securely with each other. For this purpose they need to verify one another's identity and establish a shared session key K_{AB}, although before the authentication and key distribution processing starts they do not share any secret.
- The communications between *A* and *B* have to meet potential legal requirements for warranted interception. Interception agencies in each domain are not actively involved in the authentication and key distribution procedures, but may require access to the session key K_{AB}.
- In the first scheme (Mechanism 7.3.5) a single set of TTPs, $\{T_1, ...,T_m\}$ are used as both multiple authentication servers for the users, and key escrow agencies for the interception agencies in both domains. In the second scheme (Mechanism 7.4.7) two sets of TTPs $\{T_1, ...,T_m\}$ and $\{U_1, ..., U_n\}$, one group in each domain, are used as multiple authentication servers for the users and key escrow agencies for the interception agencies. In both cases they are responsible for verifying *A*'s and *B*'s identities, establishing a session key K_{AB}, and escrowing the session key. They are trusted by both the users and interception agencies collectively, but not individually.

7.3.3.2 Mechanism 1

This escrowed key agreement scheme is based on Diffie-Hellman key exchange [98] and the verifiable secret sharing schemes described in subsection 7.3.2. In the mechanism, *A* and *B* are users in separate domains, and *m* moderately TTPs $T_1, ..., T_m$ work for both users as authentication servers, and for interception agencies in both domains as key escrow agencies. We assume that *A* and *B* have authenticated channels with T_i ($1 \le i \le m$). As in the JMW mechanism, these *m* TTPs agree a commonly held secret key $K(T_1, ..., T_n)$ and a function *f*. This function *f* shall take as input the shared secret key and the names of *A* and *B*, and generate a private integer S_{TAB}. The scheme is designed so

that for some positive integer k ($k \leq m$), any set of k TTPs can compute the session key established between A and B, but no group of k-1 or less TTPs can derive any useful information about this session key.

Mechanism 7.3.5

A set of TTPs T_1, ..., T_m assist two users A and B in establishing a session key K_{AB}, and escrow the key collectively.

1. A secretly chooses and stores its private key agreement value S_A, and computes the corresponding public value P_A (= g^{S_A}), the private shares S_{Ai} ($1 \leq i \leq m$) of S_A as defined in subsection 7.3.2.1, and the public verification sequence V_A as defined in subsection 7.3.2.2, and then sends S_{Ai} and V_A to T_i ($1 \leq i \leq m$).

2. B follows the same procedure as A, (choosing S_B, creating private shares S_{Bi}, a verification sequence V_B, and sending S_{Bi} and V_B to T_i).

3. T_i ($1 \leq i \leq m$) verifies S_{Ai}, P_A, and S_{Bi}, P_B as described in subsection 7.3.2.2. If the verification fails, T_i broadcasts the suspect share value and stops; otherwise T_i accepts the share.

4. T_i ($1 \leq i \leq m$) does the following:

   - obtains S_{TAB} by using the function f with $K(T_1, ..., T_m)$, A and B,
   - calculates P_{AT} (= $P_A^{S_{TAB}}$) and P_{BT} (= $P_B^{S_{TAB}}$), and
   - sends P_{AT} to B and P_{BT} to A.

5. A and B separately compute a session key:

$$K_{AB} = (P_{AT})^{S_B} = (P_{BT})^{S_A} = g^{S_A S_B S_{TAB}}.$$

Theorem 7.3.6

The above mechanism has the property that any group of at least k TTPs can compute K_{AB} (which is what is required for escrow purposes).

Proof

Any group of at least k TTPs can compute S_A and S_B (by the properties of the Shamir scheme discussed in subsection 7.3.2.1 above). Hence they can compute

$$K_{AB} = g^{S_A S_B S_{TAB}}$$

and the result follows.

The mechanism has been designed to make it difficult for A and B to prevent K_{AB} from being escrowed by using a hidden 'shadow-key'. In addition, no third party can force A or B to accept a wrong message unless all the third parties are colluding, and no group of fewer than k third parties can obtain any information about K_{AB}.

The method used to compose a set of key escrow agencies who are moderately trusted by mutually mistrusting domains, depends on the requirements for international secure telecommunications. The set could consist of TTPs licensed by domains other than the two domains being served, by a 'super-domain' including the two domains, or by one or other of the two domains themselves.

It would be desirable if S_{TAB} could be changed from time to time (which will mean that K_{AB} also changes). This could be achieved by including a date-stamp in the function f used to compute S_{TAB}.

Compared with a number of other proposed key establishment schemes using more than one TTP, such as, letting the two users choose the key (see [17]), letting a set of TTPs generate the key (see subsection 4.4 and [29,30]), and letting one user and two TTPs generate the key (see subsection 7.2

and [71,72,73]), this mechanism forces all involved entities, i.e. both users and the set of TTPs, to jointly generate the key, so that it may be more difficult for users and TTPs to subvert the key.

7.3.3.3 Mechanism 2

This escrowed key agreement mechanism is based on Diffie-Hellman key exchange [98] and the transferable verifiable secret sharing scheme described in subsection 7.3.2. In this mechanism, *A* and *B* are users in different domains. There are *m* TTPs T_1, ..., T_m working for A as authentication servers (in *A*'s domain), and *n* TTPs U_1, ..., U_n working for B as authentication servers (in *B*'s domain). These servers also operate as key escrow agencies for the interception agencies in their respective domains. Each set of third parties is moderately trusted by their users and interception agencies. Users and interception agencies do not communicate with TTPs outside their domain. TTP T_i ($1 \le i \le m$) can communicate with U_j ($1 \le j \le n$). Again, we assume that *A* has an authenticated channel with each T_i, and *B* has an authenticated channel with each U_j. Each group of TTPs agree a secret key $K(T_1, ..., T_m)$ or $K(U_1, ..., U_n)$ and a function *f*. This function *f* shall take as input the shared secret keys and the names of *A* and *B*, and generate private integers S_{TAB} and S_{UAB} respectively. The scheme is designed so that for some positive integer *k* ($k \le \min\{m,n\}$), any set of *k* TTPs from one or other of the two domains can compute the session key established between *A* and *B*, but no group of *k*-1 or less TTPs can derive any useful information about this session key.

Mechanism 7.3.7

Two sets of TTPs $\{T_1, ...,T_m\}$ and $\{U_1, ...,U_n\}$ assist two users *A* and *B* (respectively) to establish a session key K_{AB}. Each set of third parties escrow the key collectively.

1. *A* secretly chooses and stores its private key agreement value S_A, and computes the following values:
   - the corresponding public value P_A (= g^{S_A}),
   - the private shares S_{Ai} ($1 \le i \le m$) as defined in subsection 7.3.2.1, and
   - the public verification sequence V_A as defined in subsection 7.3.2.2,

   and then sends S_{Ai} and V_A to T_i ($1 \le i \le m$).
2. T_i ($1 \le i \le m$) verifies S_{Ai} and P_A as described in subsection 7.3.2.2. If the verification fails then T_i broadcasts the suspect share value and stops; otherwise T_i accepts the share.
3. *B* secretly chooses and stores its private key agreement value S_B, and computes the following values:
   - the corresponding public value P_B (= g^{S_B}),
   - the private shares S_{Bj} ($1 \le j \le n$) as defined in subsection 7.3.2.1, and
   - the public verification sequence V_B as defined in subsection 7.3.2.2,

   and then sends S_{Bj} and V_B to U_j ($1 \le j \le n$).
4. U_j ($1 \le j \le n$) verifies S_{Bj} and P_B as described in subsection 7.3.2.2. If the verification fails then U_j broadcasts the suspect share value and stops; otherwise U_j accepts the share.
5. T_i ($1 \le i \le m$) does the following:
   - obtains S_{TAB} by using the function *f* with $K(T_1, ..., T_m)$, *A* and *B*,
   - calculates P_{AT} (= $P_A^{S_{TAB}}$),
   - calculates S_{Aij} ($1 \le j \le n$) from S_{Ai} as defined in subsection 7.3.2.3,

- computes the 'private shares' $S_{Aij}S_{TAB}$, and their corresponding public values $g^{S_{Aij}S_{TAB}}$ as defined in subsection 7.3.2.4, and the public verification sequence V_{Ai} as defined in subsection 7.3.2.3.
- Finally, T_i sends $S_{Aij}S_{TAB}$, V_{Ai} and P_{AT} to U_j ($1 \le j \le n$).

6. U_j ($1 \le j \le n$) verifies $S_{Aij}S_{TAB}$, $g^{S_{Ai}S_{TAB}}$ and P_{AT} as described in subsection 7.3.2.3. If the verification fails then U_j broadcasts the suspect share value and stops, otherwise U_j accepts the share.
7. U_j ($1 \le j \le n$) does the following:
   - obtains S_{UAB} by using the function f with $K(U_1, ..., U_n)$, A and B,
   - calculates P_{ATU} (= $P_{AT}^{S_{UAB}}$) and sends it to B,
   - calculates P_{BU} (= $P_B^{S_{UAB}}$),
   - calculates S_{Bji} ($1 \le i \le m$) from S_{Bj} as defined in subsection 7.3.2.3,
   - computes the 'private shares' $S_{Bji}S_{UAB}$, and their corresponding public values $g^{S_{Bji}S_{UAB}}$ as defined in subsection 7.3.2.4, and the public verification sequence V_{Bj} as defined in subsection 7.3.2.3, and, finally,
   - sends $S_{Bji}S_{UAB}$, V_{Bj} and P_{BU} to T_i ($1 \le i \le m$).
8. T_i ($1 \le i \le m$) verifies $S_{Bji}S_{UAB}$, $g^{S_{Bji}S_{UAB}}$ and P_{BU} as described in subsection 7.3.2.3. If the verification fails than T_i broadcasts the suspect share value and stops; otherwise T_i accepts the share, calculates P_{BTU} (= $P_{BU}^{S_{TAB}}$), and sends it to A.
9. A and B can now separately compute the session key:

$$K_{AB} = (P_{BTU})^{S_A} = (P_{ATU})^{S_B} = g^{S_A S_B S_{TAB} S_{UAB}}.$$

Theorem 7.3.8

The above mechanism has the property that any group of at least k TTPs (in either domain) can compute K_{AB}.

Proof

The proof follows immediately from the results in subsection 7.3.2 above.

In this mechanism, the two sets of third parties in both domains do not have to trust each other, as mentioned in subsection 7.3.2.3. For the same reasons as in the previous mechanism, it is suggested that S_{TAB} and S_{UAB} should be changed as often as required.

7.3.4 Further considerations

In a key escrow system, the differing requirements of users and interception authorities are further complicated by the introduction of the key escrow agencies (or TTPs). The key escrow agencies are responsible to the interception agencies for preventing criminal users from abusing escrowed keys. Both the users and interception agencies should be in a position to check that the key escrow agencies cannot reveal escrowed keys illegally. In international key escrow, the relationships amongst these three groups of entities becomes still more complicated because more than one domain is involved. The key escrow agencies in one domain have a potential requirement to check that the key escrow agencies in the other domain cannot subvert the escrowed keys.

In this subsection, we discussion some aspects of the trust relationships between the various entities involved.

7.3.4.1 Moderately trusted third parties

There are two major reasons why we consider the use of moderately trusted third parties here.

- If interception agencies are not actively involved in session key establishment for possibly deviant users and do not store every session key themselves, key escrow agencies are required to provide a valid key when lawfully authorised. Although the key escrow agencies may not be trusted individually, a group of them might be collectively trusted by the interception agencies.
- If two users sharing no secret want to communicate security with each other, they need an authentication service provided by authentication servers. Although the servers may not be trusted individually, a group of them might be collectively trusted by their users.

Four kinds of key splitting schemes based on secret sharing schemes (e.g. [35]) have been used for splitting an escrowed key into n shares escrowed by n agencies in previously proposed key escrow systems. The first approach involves 'splitting' with an n out of n scheme, where all n components are needed to restore a given key [85]. The second approach uses splitting with an k out of n threshold scheme, which allows any subset of k of the n escrow agencies to affect the recovery of a complete key, but prohibits any group of fewer than k agencies from recovering a complete key [83]. The third approach involves splitting with an (n,t,u)-escrow scheme, which allows a subset of the escrow agencies to recover a key, where t escrow agencies could conspire without compromising a key, and n-u agencies could 'withhold' their components without interfering with key recovery ($t < u \leq n$) [82,84]. The last approach involves splitting with a 'general monotone access structure', which allows for the specification of arbitrary subsets of escrow agencies that can work together to restore a key [74].

We have used the second approach in the mechanisms described here. Actually, any one of these four splitting schemes could have been chosen, and in practice the choice would depend on the requirements for establishing a set of moderately trusted third parties. Note also that the idea of Reiter et al. regarding secure group implementation in [87] can be used to establish such a set of moderately trusted third parties and their group key.

7.3.4.2 Untrustworthy users

We now consider the case where users are not trustworthy in a key escrow system. Kilian and Leighton gave a 'shadow-public-key' attack in [83], and we now see how this attack might work in a key escrow system based on Diffie-Hellman key exchange. Each normal user generates a pair (P,S), publishes P and gives an interception authority the ability to reconstruct S, so that both the users and the interception authority can compute an escrowed key by using Diffie-Hellman key agreement. In this attack, each of two attackers instead generates two key pairs (P,S) and (P',S'), where (P,S) is a proper (public-key, private-key) pair, (P',S') is a 'shadow' key pair, and $P' = f(P)$ where f is an easily computed and publicly known function. Each of them uses (P,S) in the same way as would an ordinary user, but keeps S' reserved as his shadow private key. Both attackers separately compute a 'shadow-escrowed-key' by using his 'shadow-private-key' and the other's 'shadow-public-key'. If it is infeasible to obtain S' by knowing P, P' and S. The interception authority cannot obtain the `shadow-escrowed-key'. Furthermore the interception authority may not detect this cheating.

We presented an affine expansible verifiable secret sharing scheme in subsection 7.3.2.4, which provided the basis of users and third parties jointly generating an escrowed key in order to prevent the users from using a hidden 'shadow-key'. Note that it only makes sense to prevent criminal users from obtaining a S' which is in feasibly computed by using P, P' and S.

The further problem is how in practice to prevent criminal users from abusing the key escrow system by using improper keys, for examples, using an old escrowed key instead of a current one, using a modification of the escrowed key, e.g. which may be a publicly known function of the real escrowed key, and using a 'shadow-public-key', where S' may feasibly be computed by knowing P, P' and S. Although these abuses are all detectable, a key escrow mechanism may never check for such abuses, giving deviant users greater leeway in their abuses.

A number of approaches could be used to prevent the above abuses, such as, keeping all old escrowed keys in a valid period to check if they are used again, and monitoring all communication channels between suspected criminal users [88]. Unfortunately, these approaches may not be practical, particularly, in complicated mobile telecommunications systems.

In fact, it is impossible for a key escrow system to force two users to use only the current escrow key if the users share a secret or can use their own security system. For our purposes here, we suppose that two users, who want to communicate securely with each other, have to get assistance from key escrow agencies in order to authenticate one another's identity and establish a shared session key. We assume it is detectable if the users subvert key escrow systems by using an old escrowed key or a modified escrowed key. However we have not answered the question of how to force users to use only the current escrowed key.

7.3.4.3 Multiple mistrusting domains

So far we have discussed key escrow in mutually mistrusting domains. However, some modern secure communications may cover more than two domains. For example, in a global mobile telecommunications system, two users, respectively, are citizens of countries *C* and *D*, work for countries *E* and *F*, are registered with two mobile companies belong to countries *G* and *H*, and are roaming in two countries *I* and *J*. Their traffic might conceivably need to be intercepted by agencies in any of countries *C-J*, and hence it may be necessary to try and devise an international key escrow system which provides all governments involved with warranted access to user communications. To make matters more complicated, the countries involved may not all trust each other.

Our first mechanism, as described in subsection 7.3.3.2, could be used for this purpose. However, whether or not a set of key escrow agencies could be set up which are moderately trusted by multiple mistrusting domains, depends on political considerations beyond the scope of this discussion. The second mechanism, as described in subsection 7.3.3.3, could also be used for this purpose, at least in theory. The problem is that each set of key escrow agencies in each domain involved have to collaborate to provide contributions to the escrowed key. This may not be practical, particularly when the number of domains involved is quite large.

7.3.5 Conclusions

We have described a key escrow system using moderately trusted third parties in mutually mistrusting domains and analysed its use.

7.4 Secret key agreement using public information

7.4.1 Introduction

Wyner, [91], proposed a method for establishing a shared secret between two parties using only publicly available information. His work was innovative in that it did not make use of the widely accepted model of a secrecy system proposed earlier by Shannon, [90], whereby all parties have perfect information. Instead it relied on channel noise to enable a shared secret to be established between two parties, even though the channel was public. Subsequently, Maurer, [89], and others made use of independent public information sources and additional error-free public channels, to extend Wyner's original scheme.

This section provides an informal introduction to the subject of secret key agreement using public information. The reason why this is relevant to 3GS is that operators of future telecommunications systems will be confronted by severe key distribution problems, especially given the anticipated world-wide nature of 3GS. If they can be made practical, the ideas of Wyner and Maurer offer simple solutions to these classically difficult key management problems. Hence the main motivation for this subsection is to consider the practical problems that might arise in the utilisation of a Wyner or Maurer scheme.

In subsection 7.4.2 we review perfect secrecy, and introduce the related, but more relaxed, concept of approximate perfect secrecy required for our discussion.

In subsection 7.4.3 we show how approximate perfect secrecy can be attained when an eavesdropper's channel is merely a degraded (noisy) version of the (error-free) main channel. We then extend this to the case where the main channel is noisy, but the eavesdropper's channel is noisier still (i.e. the eavesdropper's channel is a degraded or cascaded channel). These two sections account for Wyner's original contribution to the subject.

In subsection 7.4.4 we consider the case where all channels are noisy, without the restriction that the eavesdropper's is worse, and show that approximate perfect secrecy is still possible, provided that an additional error-free public return channel is available. Subsection 7.4.5 extends these ideas to the case where the source of information is independent (i.e. it is no longer generated by one of the participants), and shows that the use of a two-way (public) error-free channel again enables approximate perfect secrecy. Subsections 7.4.4 and 7.4.5 summarise Maurer's contribution.

Finally in subsection 7.4.6 we review the results, and discuss future directions, such as the possibility of achieving practical implementations of the schemes described.

7.4.2 Perfect secrecy

Shannon defined a cipher system to be *perfect* if

$$I(M;C) = 0,$$

where M is the plaintext message and C is the ciphertext. Essentially, this means that knowledge of the ciphertext reveals no information about the plaintext.

Shannon's model of a cipher system, [90], assumes that information transmitted over public channels can be received without error by all parties. This assumption enabled Shannon to show that a message can be transmitted in perfect secrecy (i.e. the cipher is perfect) only if

$$H(K) \geq H(M),$$

where K is the key.

Wyner, [91], discarded the assumption that all parties receive transmitted information accurately, and instead considered a system whereby information is transmitted over noisy channels. When considering Wyner's approach, it is helpful to model the situation as follows: we assume there are three parties, namely, legitimate communicating entities A and B, and an eavesdropper E. A transmits a message X, whilst B and E receive messages Y and Z respectively (these are both usually noisy versions of X). We also assume that channel encoding and decoding functions e and d are utilised. Entity A encodes a message M to obtain

$$X = e(M),$$

sends X to entity B who receives Y, and decodes it to obtain

$$M' = d(Y).$$

In addition an eavesdropper E receives Z and can compute

$$M'' = d(Z)$$

(e and d are of course public). In Shannon's model, we have $X = Y = Z$ (thus $M = M' = M''$).

In our 'noisy channel' situation, the error-free receipt of information cannot be guaranteed. It is convenient to therefore introduce a slightly relaxed version of perfect secrecy which we call *approximate perfect secrecy*. This entails taking into account the channel encoding, and is defined in terms of message M' being recovered by party B with arbitrarily low error rate whilst the relative information of M and M'' is arbitrarily low.

A formal definition of approximate perfect secrecy is not required here. The concept of approximate perfect secrecy is used implicitly in the definition of *secrecy capacity*. The secrecy capacity is essentially the maximum rate at which secret information can be generated between A and B in relation to the underlying information rate of the available channels. Both Wyner and Maurer investigate the secrecy capacity of their respective schemes, though it is beyond our scope to present a detailed account here.

7.4.3 Cascaded eavesdropper channel

Suppose that the main channel between A and B is error-free, and the channel through which the eavesdropper E listens is a degraded version of this main channel.

A and B can agree on a shared secret S in such a way that E can derive negligible information concerning S. This is achieved by A transmitting randomly generated data to B (via the main public channel) whilst implementing a simple channel coding function which exaggerates the errors picked up by E.

For example, suppose A wishes to send a (randomly generated) bit b to B (throughout this document we assume all data to be binary), then b could be encoded by A with the function e_1 that assigns a randomly chosen string of n bits of the same parity as b (n is fixed, perhaps according to the BER of E's channel). At the receiver, B uses decoding function d_1 that is defined simply as the parity of the received string.

E is capable of receiving a noisy version of the string transmitted by A. The longer the string the greater the chance of this string containing errors. Thus choosing a suitably large n enables A and B to ensure that E receives negligible useful information about b. Nevertheless, B will still be able to determine b exactly.

Notice that in the example above the information rate decreases as the 'acquired secrecy' increases. One of Wyner's main results was to show that approximate perfect secrecy is possible with the information rate bounded away from zero. In fact he obtained precise bounds on the rates attainable (this is the secrecy capacity).

If the main channel between A and B is noisy and, as before, the channel through which the eavesdropper E listens in on is a degraded version of this main channel, then approximate perfect secrecy can be obtained in much the same way. The difference is that more advanced encoding and decoding functions are required. Another of Wyner's achievements was to show that suitable encoder-decoders exist.

7.4.4 General eavesdropper channel

We now show how to extend the basic idea presented in subsection 7.4.3 to the case where the channel through which E listens is not necessarily a degraded version of the main (A to B) channel. This is achieved using a further error-free public return channel (from B to A). This secondary channel can be used in two ways, as described in the following paragraphs. In both cases we assume, as before, that A generates data randomly and transmits this to B via the main channel.

The secondary channel is used by B to send the transmission he received from A, XORed with his own randomly generated data, back to A. When A receives the data from B, he removes his original data to leave the data from B, which has in effect been transmitted over the main channel. Meanwhile the only useful information the eavesdropper gains is the data from B having effectively been sent over the main channel cascaded with his own eavesdropping channel. Thus, the system reduces to the case discussed in Section

7.4.3. Notice that *B* now chooses the shared information, whilst the data generated by *A* is merely used as a mask.

Alternatively, the return channel is used by *B* to inform *A* of the bits (or words) of information received, whose reliability he is satisfied with. A measure of the reliability of the received information can be obtained through the demodulation process. *A* and *B* use only the reliable bits, discarding the rest. This effectively reduces the error rate of the main channel whilst leaving the eavesdroppers channel error characteristics unchanged (because the eavesdropper has no say in which bits of the transmission are used). Thus, we reduce the system to the case in subsection 7.4.3, where the eavesdropper's channel is inferior.

7.4.5 Independent sources

A different situation from that discussed in the previous two sections is where we have an independent source of information, and each of *A*, *B* and *E* all listening via noisy channels. To obtain approximate perfect secrecy in this case requires the use of secondary (error-free) public channels in both directions between *A* and *B*. *A* and *B* improve their effective channel error rates in just the same way as the second option described in subsection 7.4.4, by throwing away bits of the transmission they deem to be unreliably received.

We assume an independent source broadcasts randomly generated information. *A*, *B* and *E* all receive noisy versions of this. *A* and *B* use the secondary channels to communicate to each other which bits (or words) they have each received reliably. Note that, as in the previous case, the reliability of received bits (or words) might be measured through the demodulation process. If both *A* and *B* receive a particular bit reliably then it is kept and used, otherwise it is discarded. This improves the error characteristics of *A* and *B*'s channels whilst leaving the error characteristics of *E*'s channel unchanged. By XORing a number of received bits together, *A* and *B* can achieve approximate perfect secrecy is a similar manner to that described in subsection 7.4.3.

7.4.6 Summary

In subsection 7.4.3 we related Wyner's original ideas of how, given that the channel through which the eavesdropper *E* listens in on is a degraded version of the main channel between legitimate entities *A* and *B*, it is possible for *A* and *B* to agree on a shared secret *S* in such a way that *E* obtains negligible information concerning *S*. This is achieved by employing a channel encoding function which keeps the error rate of the main channel small but amplifies the errors picked up by the eavesdropper to such an extent that the amount of useful information he receives is negligible. This technique of 'condensing' information to improve secrecy is commonly known as *privacy amplification* (see for example the papers of Bennet et al., [92], [93]).

Subsection 7.4.4 extended the basic idea presented in subsection 7.4.3, by utilising a further error-free return channel (from *B* to *A*), to the case where the channel through which the eavesdropper listens is not necessarily a degraded version of the main (*A* to *B*) channel. The new error-free channel is used by *B* to send the transmission he received from *A*, XORed with his own message, back to *A*. When *A* receives this message from *B*, he can remove his original message to leave himself with a message from *B* which has in effect simply been transmitted over the main channel. Meanwhile the only useful information the eavesdropper has is the message from *B* having (effectively) been sent over the main channel cascaded with his own eavesdropping channel. Thus we reduce the system to the previous case discussed in subsection 7.4.3. Alternatively the return channel could be used by *B* to inform *A* of bits of information whose integrity he is satisfied with. This reduces the error rate of the main channel whilst leaving the eavesdroppers channel error characteristics unchanged (because the eavesdropper has no say in which bits of the transmission are used). Again, this reduces the system to the previous case where the eavesdroppers channel is inferior. The exchange of additional information between A and B to improve the quality of the channel is known as *information reconciliation*, [94], [95].

Subsection 7.4.5 addresses a slightly different situation, in which there is an independent source of random data, with each of *A*, *B* and *E* all listening in. Using error-free public channels in both directions between *A* and *B*, *A* and *B* improve their effective channel error rates by throwing away unreliable bits in just the same way as the second option discussed in subsection 7.4.4. *A* and *B* inform each other which bits they have each received with acceptable reliability. If both *A* and *B* receive a particular bit reliably then it is kept and used, otherwise it is discarded. This improves the error characteristics of the main channels to *A* and *B*,

whilst leaving *E*'s channel unchanged. Agreement of a shared secret between *A* and *B* can then easily be achieved in a manner analogous to the first case considered in subsection 7.4.3 by XORing acceptable bits together.

7.5 Secret key agreement based on storage complexity

The text in this section is based on a paper, [96], prepared for publication as part of the 3GS3 project.

We now describe a key agreement system based on the assumption that there exists a public broadcast channel transmitting data at such a rate that an eavesdropper cannot economically store all the data sent over a certain time period. The two legitimate parties select bits randomly from this channel, and use as key bits those which they have selected in common.

7.5.1 Introduction

In a recent paper, [89], Maurer has described a number of related methods for providing secret key agreement between two parties using only publicly available information. These methods are based on a development of Wyner's ideas, [91].

The particular method which has inspired the work described here relies on the two parties wishing to agree a secret key, and any eavesdropper, all receiving a signal from some channel, for which each party only receives a noisy version of the originally transmitted signal. For the system to operate, the common information derived from the channel by the two legitimate parties, *A* and *B* say, must not be a subset of the information derived from the channel by any eavesdropper. Note that it is not necessary for *A* or *B* to receive a 'better' version of the signal than the eavesdropper.

This requirement is very simply met in any situation where the channel errors are statistically independent for each of the three parties. However, in practice this requirement may be rather difficult to guarantee. As an example consider a radio channel: the eavesdropper might be able to position his antenna close to one of the legitimate parties, and hence obtain a strictly better version of the signal than one of the legitimate parties.

It is also necessary for the parties *A* and *B* to share an error-free 'authenticated channel', although this may also be intercepted by the eavesdropper. By an authenticated channel we mean one in which *B* is able to verify that all the data received on the channel originated from *A* in exactly the same form as was received, and vice versa. This may also be non-trivial to provide in practice.

We consider a slightly different key agreement system, which avoids the first requirement above. This system is based on the assumption that there exists a public broadcast channel transmitting data at such a rate that an eavesdropper cannot economically store all the data sent over a certain time period. The two legitimate parties randomly choose which bits to store from this broadcast channel, and then compare notes (using an authenticated channel which need not provide privacy) after the chosen time period has passed as to which bits they have both selected, which then constitute the shared secret key. Unless the eavesdropper has stored a high proportion of all the bits transmitted on the public broadcast channel, it will almost certainly have no more than a small proportion of the secret key bits.

This idea is analogous to the encryption system described by Maurer in [97], where the idea of a noise source producing data in quantities which cannot economically be stored is also exploited. However, the idea in [97] is rather different, in that the legitimate users do not divulge which of the bits they have used.

In the scheme described here the provable guarantees of security which can be obtained for Maurer's schemes, [89], are thus exchanged for arguments regarding the cost (in providing large amounts of data storage) to a third party of obtaining the key agreed by *A* and *B*. It is important to note that this scheme, like all the schemes in [89], does still require the error-free authenticated channel.

Before proceeding observe that, for clarity and brevity of presentation, we only provide informal arguments to support certain of our main results. Formal proofs can be constructed using information theoretic arguments.

7.5.2 The basic scheme

In order to describe our key agreement system we first need to describe our model of the communicating parties. We suppose that *A* and *B*, the two legitimate parties wishing to establish a

shared secret key, both have access to an (error free) public broadcast channel sending random binary data at a high rate, say R bits/sec. We suppose also that A and B:

- share an authenticated error-free channel,
- have the means to store n_A and n_B bits respectively, and
- are 'synchronised' with respect to the broadcast channel, i.e. they have the means to refer to a single bit sent on this channel.

By an authenticated channel shared by A and B we mean a channel for which A can be sure that any bits received claiming to be from B are genuinely from B, and have not been manipulated in transit (and vice versa).

The eavesdropper, who we call C, also has error-free access to both the public broadcast channel and the authenticated channel between A and B.

The basic key agreement system now works as described in the following procedure. Note that we subsequently describe some improvements on this basic scheme.

Algorithm 7.5.1

1. A and B both monitor the broadcast channel for an agreed interval of time of duration T. The start and end points of this interval can be agreed using the authenticated channel. We assume that the exact details of the time interval are also known to the eavesdropper C.
2. During this interval A selects n_A of the bits sent over the broadcast channel at random and stores them. Similarly, and independently, B selects n_B bits at random and stores them.
3. At the end of the interval (and not before) A sends B the 'indices' of the bits that it has stored, where we assume that the bits sent over the public channel during the time interval are successively given the indices 0,1,..., etc. Having received these indices, B examines them and compares them with the indices of the bits it has stored and makes special note of any coincidences.
4. B now sends back to A a list of all coincidences, and the bits corresponding to these coincident indices become the key bits (which both A and B have).

As we now show, given that A and B make genuinely random selections, and T, R, n_A and n_B are chosen appropriately, the eavesdropper will be obliged to store many more bits than either A or B to have a good probability of knowing more than a very few of the key bits. To demonstrate this we first establish the following simple result.

Theorem 7.5.2

Suppose A and B follow Algorithm 7.5.1, and an eavesdropper stores n_C bits randomly selected from the public broadcast channel during the agreed time interval. Suppose also that n_A and n_B are both very much smaller than TR (the number of bits sent during the agreed time interval). Then the following will hold.

i. The expected number of bits of key shared by A and B at the end of the process will be approximately $n_A n_B/TR$.

ii. The expected number of key bits available to C will be approximately $n_A n_B n_C/(TR)^2$.

Proof (Sketch)

Both results follow from elementary probability considerations.

i. For any given bit of the n_A selected by A, the probability that it is also selected by B is n_B/TR. Given that n_A is very small with respect to TR, we may ignore the fact that the probabilities are not independent and hence say that the expected size of the set of bits selected by both A and B will be *approximately* n_A times the above probability, and (i) follows.

ii. This follows by a precisely analogous argument.

Before proceeding note that A will actually need $(L+1)n_A$ bits of storage, where $L = \lceil \log_2(\mathrm{TR}) \rceil$, i.e. L is the number of bits in the index values. Similarly B will need $(L+1)n_B$ bits of storage. In addition A will need to send Ln_A bits over the authenticated channel as part of the key agreement process.

We have thus converted a theoretically secure system into one which provides a cost difference between legitimate key agreement and unauthorised interception of key material.

7.5.3 A simple example of the system

To illustrate how such a system might operate we consider a simple example.

Suppose T is 10^5 seconds (i.e. approximately one day) and R is 10^{10} bits per second, i.e. 10 Gbits/sec, and hence $TR=10^{15}$ and $L=50$. Now suppose that $n_A=n_B=3\times10^8$ and hence A and B will need to have $51\times3\times10^8$ bits of storage, i.e. a little under 2 Gigabytes. At current prices, high speed magnetic disk storage of this capacity will cost approximately £1000, and prices are likely to continue to fall. At the same time, A will need to send a little under 2 Gigabytes of information to B over the shared authenticated channel. By Theorem 7.2, at the end of the process A and B will expect to share a key of approximately $(3\times10^8)^2/10^{15}=90$ bits.

We next consider what strategy the eavesdropper might adopt to try and learn significant amounts of key information. In order to obtain, say, 10% of the key bits, by Theorem 7.5.2 the eavesdropper will need to store 10% of the bits sent over the public broadcast channel. This will require 10^{14} bits of storage, i.e. approximately 12,000 Gbytes. At today's prices, low cost storage (e.g. magnetic tape) still costs significantly more than £10 per Gbyte, and hence such storage will cost the eavesdropper well in excess of £120,000, and, by similar arguments, to obtain 50% or 100% of the key bits would cost in excess of £600,000 or £1,200,000 respectively.

If A and B were concerned about the possibility that an eavesdropper could make use of a small number of the key bits, then security could be increased by using a one-way hash function to produce, say, a 64-bit key from the 90 bits derived from the key exchange process.

7.5.4 Extensions of the basic scheme

There are a number of ways in which the basic system can be modified to increase the cost differential between the legitimate parties and the eavesdropper. We consider two such possibilities. Both offer methods by which the storage requirements for A and B can be reduced from $(L+1)n_A$ and $(L+1)n_B$ to around n_A and n_B respectively.

7.5.4.1 Pseudo-random selection of bits by A *and* B

Suppose A and B have agreed in advance the choice of a cryptographically secure pseudo-random number generator. By this we mean a generator which, given a secret key as input, produces a sequence of pseudo-random numbers as output and for which, given knowledge of some of the output sequence, it is computationally infeasible to compute any more information regarding the output sequence. In particular this implies that, given knowledge of some of the output sequence, it will be computationally infeasible to deduce the key used to generate this sequence.

We now describe a modified version of the basic key agreement system described in Algorithm 7.5.1, which makes use of these pseudo-random number generators.

Algorithm 7.5.3

1. Before starting the process A and B select random keys for their chosen pseudo-random number generator, which we call R_A and R_B respectively.

2. A and B both monitor the broadcast channel for an agreed interval of time of duration T. The start and end points of this interval can be agreed using the authenticated channel. We assume that the exact details of the time interval are also known to the eavesdropper C. For

the purposes of this discussion we suppose that the bits sent over the broadcast channel during the selected time interval are labelled

$$b_0, b_1, \ldots, b_{TR-1}.$$

3. Before starting the monitoring *A* and *B* also choose *step values* s_A and s_B respectively, where $s_A = \lfloor RT/n_A \rfloor$ and $s_B = \lfloor RT/n_B \rfloor$. If necessary, at some point (either before, during or after the monitoring period) *A* and *B* exchange these step values.

4. During the time interval, *A* uses the chosen pseudo-random number generator and its secret key R_A to produce a sequence $t_0,t_1,\ldots,t_{nA-1}$ of pseudo-random numbers, where each pseudo-random number t_i is chosen from the range $0,1,\ldots,s_A-1$ (with uniform probabilities). *A* then selects the following n_A bits sent over the broadcast channel and stores them:

$$b_{t_0}, b_{s_A+t_1}, b_{2s_A+t_2}, \ldots, b_{(n_A-1)s_A+t_{n_A-1}}.$$

Similarly, and independently, *B* uses the pseudo-random number generator and its secret key R_B to produce a sequence $u_0,u_1,\ldots,u_{nB-1}$ of pseudo-random numbers, where each pseudo-random number u_i is chosen from the range $0,1,\ldots,s_B-1$ (with uniform probabilities). *B* then selects the following n_B bits sent over the broadcast channel and stores them:

$$b_{u_0}, b_{s_B+u_1}, b_{2s_B+u_2}, \ldots, b_{(n_B-1)s_B+u_{n_B-1}}.$$

5. At the end of the interval (and not before) *A* sends *B* its secret key R_A, used to help select which bits from the public channel it has stored during the time interval. Similarly *B* sends *A* its secret key R_B.

6. *A* can then use R_A and R_B to find those values of *i* and *j* ($0 \le i < n_A$, $0 \le j < n_B$) for which $is_A+t_i = js_B+u_j$. This can be done with the minimum of storage by at any point retaining the values of i, t_i, j and u_j, and then

   - replacing the pair (i,t_i) with the pair $(i+1,t_{i+1})$ if $is_A+t_i < js_B+u_j$,
   - replacing the pair (j,u_j) with the pair $(j+1,u_{j+1})$ if $is_A+t_i > js_B+u_j$, and
   - storing the values $(i-1)s_A+t_i$ whenever $is_A+t_i = js_B+u_j$.

7. *B* can do precisely the same calculations, leaving *A* and *B* with a known set of mutually held bits (which can be used to create a key).

Before attempting to describe the performance of this modified scheme, we first consider the best strategy for an eavesdropper wishing to find as many key bits as possible using the minimum amount of storage. There would appear to be three obvious strategies for the eavesdropper, *C*. Firstly *C* could choose to store a random selection of bits from the channel (without regard to the 'step values'). Secondly *C* could store a fixed number of bits from each range,

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

for $t = 0,1,\ldots,n_A-1$. Thirdly, *C* could select a number of ranges

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

for various values of t, ($0 \le t < n_A$), and store all the bits for the selected ranges. Note that, alternative versions of the second and third strategies would involve replacing s_A and n_A with s_B and n_B.

Now, given that there is at most one key bit within any range

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

the second strategy would seem to be the best (although all strategies yield very similar results when n_A and n_B are small relative to TR). In the following simple result, in which we derive the

performance of this revised scheme, we therefore assume that the eavesdropper is using the second of the above strategies.

Theorem 7.5.4

Suppose *A* and *B* follow Algorithm 7.5.3, and an eavesdropper *C* stores n_C bits selected from the public broadcast channel during the agreed time interval. Suppose also that $n_C=dn_A$ for some integer *d*, and that *C* stores *d* bits from each range

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

($0 \le t < n_A$). As previously we assume that n_A and n_B are both very much smaller than *TR* (the number of bits sent during the agreed time interval). Then the following will hold.

i. The expected number of bits of key shared by *A* and *B* at the end of the process will be approximately n_An_B/TR.

ii. The expected number of key bits available to *C* will be approximately $n_An_Bn_C/(TR)^2$.

Proof (Sketch)

i. Consider any range:

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

(for some value of *t* satisfying $0 \le t < n_A$). At the end of the agreed time interval, *A* will store exactly one bit from this range. The probability that *B* will also store this bit is equal to n_B/TR. Given that there are n_A such ranges, and assuming that these probabilities are independent (which is a reasonable approximation given n_A and n_B are small with respect to *TR*), we see that the expected number of bits held by both *A* and *B* at the end of the agreed time interval is approximately equal to n_An_B/TR, as required.

ii. As previously, consider any range:

$$b_{ts_A}, b_{ts_A+1}, \ldots, b_{(t+1)s_A-1}$$

(for some value of *t* satisfying $0 \le t < n_A$). At the end of the agreed time interval, *A* will store exactly one bit from this range. The probability that *B* and *C* will also store this bit is equal to $(n_B/TR)(d/s_A)$. Given that there are n_A such ranges, and assuming that these probabilities are independent (which is a reasonable approximation given n_A and n_B are small with respect to *TR*), we see that the expected number of bits held by all of *A*, *B* and *C* at the end of the agreed time interval is approximately equal to $n_An_Bd/TRs_A = n_An_Bn_C/T^2R^2$, as required.

Before proceeding note that *A* will only need n_A bits of storage. Similarly *B* will only need n_B bits of storage. In addition *A* and *B* will only need to send their respective secret keys R_A and R_B over the authenticated channel as part of the key agreement process.

Hence this amended procedure reduces the storage for *A* and *B* to n_A and n_B respectively, minimises the use of authenticated channel, and also makes the match-finding process a simple one. This is at the cost of making the security dependent on the computational security of the pseudo-random number generator(s) employed by *A* and *B*. To show how effective this improvement is we consider a modified version of our previous example; we use the same cost assumptions as in subsection 7.5.3.

Suppose *T* is 10^5 seconds (i.e. approximately one day) and *R* is 10^{12} bits per second, i.e. 1000 Gbits/sec, and hence $TR=10^{17}$. Now suppose that $n_A=n_B=3\times10^9$ and hence *A* and *B* will need to have 3×10^9 bits of storage, i.e. a little under 400 Megabytes, costing no more than £200. At the same time, *A* will need to send only one key (of say 128 bits) to *B* over the shared authenticated channel (and vice versa). By Theorem 7.5.4, at the end of the process *A* and *B* will expect to share a key of approximately $(3\times10^9)^2/10^{17}=90$ bits.

We next consider the position of the eavesdropper. In order to obtain, say, 10% of the key bits, the eavesdropper will need to store 10% of the bits sent over the public broadcast channel. This will require 10^{16} bits of storage, i.e. approximately 1,200,000 Gbytes. Such storage will cost the eavesdropper well in excess of £12,000,000, and, by similar arguments, to obtain 50% or 100% of the key bits would cost in excess of £60,000,000 or £120,000,000 respectively.

7.5.4.2 Block-wise selection of bits

In the previous section we described a system which minimises both the storage requirements for *A* and *B* and the use of the authenticated channel for *A* and *B*. This was at the cost of making the security depend on the cryptographic properties of a pseudo-random number generator. We now consider a slightly different modification of the basic scheme which retains many of the advantages of the scheme described in subsection 7.5.4.1, but which does not rely on any computational security assumptions.

The procedure is as follows.

Algorithm 7.5.5

1. *A* and *B* both monitor the broadcast channel for an agreed interval of time of duration *T*. The start and end points of this interval can be agreed using the authenticated channel. We assume that the exact details of the time interval are also known to the eavesdropper *C*. For the purposes of this discussion we suppose that the bits sent over the broadcast channel during the selected time interval are labelled

   $$b_0,b_1,\ldots,b_{TR-1}.$$

   To make our discussions simpler we also assume that $n_A|TR$ and $n_B|TR$, and hence define s_A and s_B by $s_An_A = s_Bn_B = TR$. Suppose moreover that $s_As_B|TR$, and define w by $s_As_Bw = TR$. We assume that all these parameters are known to *A* and *B* before the start of the agreed time interval.

2. During the time interval *A* randomly chooses w values $p_0,p_1,\ldots,p_{w-1}$, where each value p_i satisfies $0\le p_i<s_A$. *A* then stores the following w sets of s_B bits during the agreed time interval (i.e. a total of n_A bits):

   $$b_{is_As_B+p_is_B}, b_{is_As_B+p_is_B+1}, \ldots, b_{is_As_B+p_is_B+s_B-1}$$

   for i=0,1,...,w-1.

   Similarly, and independently, *B* randomly chooses w values $q_0,q_1,\ldots,q_{w-1}$, where each value q_i satisfies $0\le q_i<s_B$. *B* then stores the following w sets of s_A bits during the agreed time interval (i.e. a total of n_B bits):

   $$b_{is_As_B+q_i}, b_{is_As_B+s_B+q_i}, \ldots, b_{is_As_B+(s_A-1)s_B+q_i}$$

   for i=0,1,...,w-1.

3. At the end of the agreed time interval it should be clear that *A* and *B* will share precisely one bit from each range of bits

   $$b_{is_As_B}, b_{is_As_B+1}, \ldots, b_{(i+1)s_As_B-1}$$

   for i=0,1,...,w-1. I.e. *A* and *B* will share precisely w bits.

4. At the end of the agreed time interval (and not before) *A* sends *B* its random values $p_0,p_1,\ldots,p_{w-1}$, used to help select which bits from the public channel it has stored during the time interval. Similarly *B* sends *A* its secret random values $q_0,q_1,\ldots,q_{w-1}$.

5. *A* and *B* can then both very easily determine which key bits they share.

As in the previous section, before attempting to describe the performance of this scheme, we first consider the best strategy for an eavesdropper wishing to find as many key bits as possible using the minimum amount of storage. There would appear to be three obvious strategies for the eavesdropper, *C*. Firstly *C* could choose to store a random selection of bits from the channel. Secondly *C* could store a fixed number of bits from each range,

$$b_{ts_As_B}, b_{ts_As_B+1}, \ldots, b_{(t+1)s_As_B-1}$$

for t=0,1,...,w-1. Thirdly, *C* could select a number of ranges

$$b_{ts_As_B}, b_{ts_As_B+1}, \ldots, b_{(t+1)s_As_B-1}$$

for various values of t, ($0 \le t < w$), and store all the bits for the selected ranges.

Now, given that there is exactly one key bit within any range

$$b_{ts_As_B}, b_{ts_As_B+1}, \ldots, b_{(t+1)s_As_B-1}$$

($0 \le t < w$), all strategies would appear to yield similar results. In the following simple result, in which we derive the performance of this revised scheme, we therefore arbitrarily assume that the eavesdropper is using the second of the above strategies.

Theorem 7.5.6

Suppose *A* and *B* follow Algorithm 7.5.5, and an eavesdropper *C* stores n_C bits selected from the public broadcast channel during the agreed time interval. Then the following will hold.

i. The number of bits of key shared by *A* and *B* at the end of the process will be exactly $w = n_An_B/TR$.

ii. The expected number of key bits available to *C* will be $wn_C/TR = n_An_Bn_C/(TR)^2$.

Proof (Sketch)

i. This follows from the discussion given as part of Algorithm 7.5.5.

ii. Consider any range:

$$b_{ts_As_B}, b_{ts_As_B+1}, \ldots, b_{(t+1)s_As_B-1}$$

(for some value of t satisfying $0 \le t < w$). At the end of the agreed time interval, *A* and *B* will store exactly one bit from this range. The probability that *C* will store this particular bit is equal to n_C/TR. Given that there are w such ranges, and given that these probabilities are independent, we see that the expected number of bits held by all of *A*, *B* and *C* at the end of the agreed time interval is equal to wn_C/TR, as required.

This system then achieves a comparable performance to the system in the previous section. |the only disadvantage is the slightly increased communication cost in transferring the values p_i and q_i across the authenticated channel. However, this is a very small cost since the number of these values will only be the same as the number of key bits agreed by *A* and *B*.

Moreover this variant of the basic scheme has two significant advantages.

- Unlike the first variant (described in Section 7.5.4.1), its security is not dependent on the cryptographic properties of a pseudo-random number generator.
- Unlike both the other schemes, it yields a key of guaranteed length to *A* and *B*, and not just a varying number of key bits with an associated expected value. The disadvantage of this latter case is that on some occasions the number of key bits provided to *A* and *B* may be somewhat less than the expected value, potentially causing problems.

7.5.5 Summary and conclusions

We have thus described systems which provide secret key agreement between *A* and *B* and whose security rests solely on the following two assumptions.

- The cost of storage remains high relative to the bandwidth of one or more publicly available broadcast channels.
- *A* and *B* share an error-free authenticated channel.

It is particularly interesting to note that, with the exception of the scheme described in Section 7.5.4.1, the system's security does not depend on any assumptions regarding the computational difficulty of any problems. In that sense the systems are provably secure (given the two key assumptions listed above).

We now briefly consider possible practical circumstances in which the above two assumptions might be satisfied.

There are various ways in which the two legitimate parties might be provided with an authenticated channel (but not with the means to securely agree a key). Two of the more likely are as follows.

- The users may purchase a communications facility which provides an authenticated channel as a premium service (which can be obtained simply by paying the appropriate rate). The communications service provider may, for example, provide this by using digital signatures, MACs or some other type of cryptographic check function. It is certainly conceivable that this could be provided in such a way that the users have no access to the keys used, and hence no direct means to exchange secret keys.
- The users may have access to an implementation of a digital signature function such as DSS, which cannot be used for data encryption. They could then use this digital signature function, in conjunction with authenticated keys for each other, to provide the authenticated channel.

We next briefly describe two possible sources for a high bit rate (say greater than 10 Gbit/sec) public broadcast channel.

- The first is to make use of a high rate public satellite data channel. In this case the bits sent over the channel will not be random---instead they will consist of many data streams intermingled. However, in practice they may be 'random enough' for our purposes (especially if a hashing operation is performed on any agreed set of key bits).
- The second is to employ a purpose-designed high speed random data source, the output of which is made publicly available by some means (e.g. by fibre-optic cable). Although this will now guarantee randomness for the bits, there are obvious problems with this approach if it is simultaneously used by many pairs of parties to agree a secret key (as would almost certainly be necessary to justify the cost of providing such a channel). In such an event there are much greater incentives for third parties to invest resources in storing the entire channel output, since it could potentially yield many secret keys simultaneously.

Finally, it could be argued that, given that *A* and *B* share an authenticated channel, then they can achieve secret key agreement by using the well-known Diffie-Hellman key exchange protocol, (see, for example, [98]). Whilst this is certainly true, there may be situations where users do not wish to take such an approach. For example, users may not choose to trust a system whose security depends entirely on a single mathematical function remaining hard to compute (i.e. the discrete logarithm problem), and they may prefer to trust in arguments about the likely cost of data storage. It is certainly of theoretical interest to observe that key agreement schemes can be devised which rely only on the two assumptions listed above, and which do not require any assumptions about the computational difficulty of certain calculations.

8. Terminal-related security

8.1 Introduction

We now consider the somewhat thorny issue of terminal security. This issue can be divided into two main problems:

- What level of terminal security should be provided (if any)?
- How should any identified security features be implemented, bearing in mind the need to minimise the network overhead?

We start this discussion in subsection 8.2, where we briefly summarise the possible security threats to user terminals.

In subsection 8.3 we take this analysis one step further by briefly discussing some of the options for providing terminal security features.

Note that a detailed study of one possible approach to the terminal security problem can be found in [99]. This technical report, prepared as part of the 3GS3 project, is too long and specialised to include here. However, copies can be obtained by writing to: *Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England* (quoting the title and number of the report).

8.2 Terminal-related security

This brief section outlines certain security threats to 3GS terminals, and indicates possible solutions. This is not an exhaustive account. The role of this section is to serve as a preface to subsequent discussions.

8.2.1 Theft

A simple solution to this threat is for owners to look after their equipment, and perhaps get it insured. If equipment is stolen then its theft can be reported to the appropriate authorities.

If the equipment is valuable, then it might have a physically secured identity which can be easily read but not changed. This would assist authorities in recognising the equipment.

Basic equipment will be very cheap, so it would not be a great loss and, in any case, would probably not be worth stealing.

8.2.2 Non-type approval

The proposed 'mass market' will ensure the low cost of basic equipment. A manufacturer will find it very hard to produce reasonably priced equipment, even if it is sub-standard, unless he is mass-producing it. Given that he is that serious, then he may as well seek type approval, which will not be difficult. Taking extreme action against offenders will also discourage would-be offenders.

8.2.3 Cloning

This is used to disguise stolen or non type approved equipment, so it becomes fairly pointless given that these threats are themselves acceptably small.

8.2.4 Defective equipment

This could threaten users (e.g. microwave radiation causing tissue damage), third parties (e.g. interfering with hearing aids), and providers (e.g. excessive power causing interference).

Causing channel interference might be considered serious by providers. The threats to human users and third parties should also be treated seriously (one fatality could seriously affect a manufacturer's credibility and share price).

Type approval and equipment barring could be used to combat this threat.

8.2.5 Abuse of terminals

Terminals could be used as unauthorised listening devices, or could perhaps be altered to enable them to 'trap' sensitive user data.

Threats related to the use of pay-phones and other 'visited terminals' (e.g. providing services to other parties on a user's account) must also be considered.

8.3 Addressing mobile equipment security

8.3.1 Introduction

The subject of mobile equipment (ME) security for 3GS has been sadly neglected. Although a number of initial studies have taken place, no formal proposals have been made as to how security requirements on MEs should be addressed. Until specific solutions are put forward, it is unlikely that further progress will be made. In view of this, we attempt to identify the important factors regarding ME security, and make preliminary proposals for suitable mechanisms.

8.3.2 Factors

The overriding factors regarding ME security are as follows:

- centralised EIRs holding whitelists of MEs is inconceivable in a global system;
- an open retail market for MEs is essential;
- authorities will not allow the unregulated manufacture of MEs;
- the risk of injury to humans and animals through the effects of sub-standard or faulty MEs is intolerable;
- the risk of injury to humans through violence associated with theft of MEs is intolerable;
- adverse effects on profits (to parties involved in the provision of mobile communications) through loss of user confidence in the system security (and the corresponding reduction in subscriber numbers) is undesirable.

8.3.3 Potential Solutions

8.3.3.1 Dual local/network based detection mechanism

Assume the ME has a secret and public key pair, and a certificate (public key + IMEI signed by type approval authority). A UIM can then authenticate the ME locally without recourse to the network (UIM checks the ME's certificate and then authenticates the ME by some form of challenge-response). In addition, the UIM can send the IMEI to the network (to an EIR) for checking against a blacklist (containing stolen, faulty and otherwise revoked MEs).

The local authentication could be carried out each time the UIM is inserted into the ME. The network based check should occur less frequently, perhaps just the first time a particular ME is used with a UIM. In this case the UIM is required to keep a short list of the most recent MEs it has been used with.

Notes:

- no whitelists are required to be kept by EIRs;
- an open market for MEs is still possible (no initialisation or registration of MEs beyond type approval is necessary);
- signalling of ME related data over the air and within networks (between VLR and EIR) will be infrequent (it will depend upon how often a user uses different MEs);
- rechipping (cloning) MEs with key pairs taken from (known) stolen MEs is detectable (the act of `rechipping' still appears to be far too easy in current systems);
- rechipping MEs with key pairs generated by criminals is prevented (the keys cannot be certified).
- the mechanism admits the subsequent tracking and/or barring of MEs;

- use of the mechanism must be mandatory, and should not be offered as a supplementary service. If it were, then criminals could undermine the mechanism simply by not subscribing to it.;
- EIRs and a CEIR are required (for blacklisting);
- a centralised type approval process is required to ensure that there are few certification authorities (authority's public keys must be stored in the UIM);
- rechipping with key pairs stolen from MEs not registered as stolen is undetectable (e.g. the criminal uses his own ME key pair - if he can extract them).

8.3.3.2 Local deterrent

Purely local mechanisms (i.e. those carried out wholly on the mobile side) are likely to be less powerful than network based mechanisms (they can be no more than deterrents), but have the advantage that they are easier to implement and use less resources. Local mechanisms could be incorporated in addition to a network based solution. One technique is to 'lock' MEs to UIMs or human users. Three instances of this approach are as follows:

- insert some UIM specific data into the ME, and have ME - UIM authentication prior to use;
- remove some vital data from the ME (e.g. its certificate) and store it in the UIM whilst locked;
- use an access control mechanism similar to that used for user-UIM access (this is UIM independent, so the owner can still unlock their ME without having their UIM available).

9. Identity and location privacy

9.1 Introduction

In mobile telecommunications systems, each user must let his or her Service Provider know where he/she is so that a call route can be maintained to the user. This is achieved by the use of registration and location update mechanisms, by means of which the user provides his/her Service Provider with location information, via a Network Operator for the current location area. This has the side effect that anyone wishing to track this particular user can do so by monitoring the identity and location messages transmitted during these registration and location update procedures. In order to protect users' identity and location information against disclosure to unauthorised entities, an Identity and Location Privacy mechanism is needed.

In GSM, Identity and Location Privacy is provided by the use of Temporary Identities (or *TMSIs - Temporary Mobile Subscriber Identities* as they are called in GSM). These TMSIs are sent over the air interface instead of the real user identity. Moreover, these TMSIs are updated regularly in an untraceable way. Unfortunately, for operational reasons the real identities need to be transmitted over the air interface from time to time. Worse still, a third party masquerading as a Base Station can, at any time, force a mobile to transmit its real identity.

The purpose of this section is to identify new Identity and Location Privacy mechanisms appropriate to third generation systems, which do not contain the shortcomings of some second generation systems, and which are not incompatible with the requirements for calling line identity features and for lawful access to call-related information.

In subsection 9.2 the problem is discussed in some detail, and an attempt is made to classify approaches to solving the problem. An example scheme is also given.

In subsection 9.3 a combined authentication and identity/ location privacy mechanism is described, which is based on the use of symmetric cryptography. The design can be regarded as a natural evolution of the ETSI DECT authentication scheme.

In subsection 9.4 another combined authentication and identity/location privacy mechanism is described, this time based on the use of asymmetric cryptography (more specifically, it relies only on the use of public key encipherment). A detailed evaluation of this mechanism is provided in subsection 9.5.

9.2 ILP in UMTS

The text in this section is based on a paper, [100], prepared for publication as part of the 3GS3 project.

9.2.1 Introduction

In mobile telecommunications systems, each user must let its Service Provider (SP) know where he/she is so that its call route can be maintained by the system. This is achieved by the registration and location update mechanisms which the user uses to tell its current location to the SP via a Network Operator (NO) for the current location area. This has the side effect that anyone wanting to track this particular user can do so by monitoring the identity and location messages transmitted during the registration and location update processes. In order to protect users' identity and location information from being made available or disclosed to unauthorised entities or processes, an Identity and Location Privacy (ILP) mechanism is needed. This mechanism protects users against tracing of their physical location by illegal means.

The purpose of this part of the report is to give a general discussion of ILP mechanisms in third generation mobile telecommunications systems (3GS). The discussion begins in the next subsection with a brief overview of the approach to implementing ILP supported by GSM, and is followed by an analysis of possible threats to this approach in subsection 9.2.3. In subsection 9.2.4 we consider the provision of ILP in a more general context. In subsection 9.2.5 a protocol for service related authentication providing ILP is proposed. We conclude in subsection 9.2.6.

9.2.2 The GSM approach

In GSM, ILP is achieved by using Temporary Identities (TIs, denoted in GSM by temporary mobile subscriber identities or TMSIs) over the air interface instead of Real Identities (RIs, denoted in GSM by international mobile subscriber identities or IMSIs). The value of the TI is chosen by a NO (denoted in GSM by BSS/MSC/VLR, i.e. Base Station Subsystem, Mobile Switching Centre and Visitor Location Register) and transmitted to the user in encrypted form. The TI is a local identity and valid only in a given location area.

The SP (denoted in GSM by AuC/HLR, i.e. Authentication Centre and Home Location Register) maintains a database of the current TI/RI relationships and is therefore able to determine the real identity of the user, i.e. it can determine the RI from the TI. TIs are changed on each location update and certain other occasions as defined by the network. The user identifies himself by sending the old TI during each location update process. The NO then allocates a new TI and returns it. The old TI has to be sent before authentication takes place, and must therefore be sent unencrypted.

However, the new TI is returned after authentication is completed and a new session key has been generated. Therefore the new TI can be, and is, encrypted before it is returned to the user. If the TI is not available or is invalid (e.g. if during the initial location registration the old NO is not reachable, or the old TI is unknown, [40], then the user has to identify itself using its RI. In this event a new TI is allocated and returned encrypted.

9.2.3 Possible threats to the GSM approach

In this section we consider seven possible threats to the GSM ILP scheme:

T1. **Intercepting communications between a user and NO**. An intruder is able to obtain a RI from the GSM air interface in the following three cases when a user sends the RI in clear text:

- initial location registration,
- old visitor location register is not reachable, and
- no old TI is available.

T2. **Impersonating a user**. In a mobile telecommunications environment it may be possible for an intruder to fabricate and/or interfere with a user's messages to an NO. An intruder could modify

the user's TI and/or the Location Area Identifier (LAI), both of which are sent from the user to the NO in clear text. The result of such an action will mean that the NO does not recognise the user (or it cannot contact the 'old' NO) which will, in turn, cause the NO to request the user to send its RI unencrypted over the Air Interface. Such a procedure could be repeated as often as required, enabling a third party to track a user.

T3. **Impersonating an NO**. In GSM the user authentication process is unilateral, i.e. the NO verifies the identity of the user, but the user does not verify the identity of the NO. Hence a third party could impersonate an NO and send a message to a user requesting it to send its RI unencrypted over the Air Interface. As is the case for threat T2, such a procedure could be repeated as often as required, enabling a third party to track a user.

T4. **Intercepting channels between NOs and SPs**. It may be possible for an intruder to observe a user's identity and location information and hence track this user by intercepting channels between NOs and SPs, because each updated location message needs to be sent from an NO to an SP, possibly in clear text.

T5. **Malicious NOs**. It is possible for a malicious NO to track a user because TIs are chosen by NOs, and hence NOs have access to a user's RI.

T6. **Impersonating an SP to a NO**. In GSM the SP verifies the identity of the user during the user authentication process, but no mechanisms are provided for the NO and/or the user to verify the identity of the SP. Of course, in practice where such a threat exists proprietary techniques are used to protect SP/NO communications, and hence (indirectly) protect the user against a third party impersonating an SP. However, if the NO does not authenticate the SP, and a threat exists, then an intruder could impersonate an SP to an NO to obtain the user's identity and location information (and thereby track the user).

T7. **Malicious SPs**. A user's physical location could be disclosed to unauthorised processes if its SP abuses the user's identity and location information. However, it is essential that the SP knows the user's identity and location since the user has a contractual/charging relationship with its SP; hence we do not address this threat further here. However, SPs will need to take very careful steps to protect their users against breaches of privacy. Hence SPs will typically need to have their employees' access to client information both limited and audited, i.e. SPs will need to utilise secure access control mechanisms for their user databases.

9.2.4 Providing identity and location privacy

We now discuss the provision of Identity and Location Privacy in a more general context.

9.2.4.1 Requirements for an ILP mechanism

We start by listing possible general requirements for an ILP mechanism (based on our analysis of the GSM scheme).

- The user's RI should never be transmitted unprotected across the air interface (hence addressing threats T1, T2 and T3).
- The user's RI should never be transmitted unprotected between network entities (NOs and/or SPs), unless the communications path is inherently secure (hence addressing threat T4).
- The user's RI should only be divulged to those parties who need it for correct network operation; in the limit this could mean that the user's SP is the only entity who possesses the user's RI (hence addressing threat T5). It would seem that only the user's SP needs to know the user's RI and not the NO, since when an NO provides service to a user it does not need to know what the user's real identity is. The NO only needs to know who the user's SP is, and have an identifier so that the NO subsequently has the means to charge the SP for service provided to the user. In such a circumstance the SP will need to preserve this identifier so that the charge can be matched against the user's true identity.

- Third parties should not be able to track users by masquerading as an SP to an NO, as an NO to a user, as a user to an NO, or as an NO to an SP (hence addressing threats T2, T3 and T6).

Clearly not all these requirements will be capable of being met in a practical system. However, it does not seem unreasonable to expect at least the first requirement to always be met (although GSM does not even meet this requirement).

9.2.4.2 General approaches for providing ILP

We now discuss two general approaches for meeting the ILP requirements identified immediately above. The provision of ILP is typically combined with the provision of entity authentication, and both the approaches we discuss assume that this combination will take place.

The fundamental problem is to meet the first identified requirement, i.e. to avoid the transmission of users' RIs across the air interface. First observe that the reason why addresses (of some kind) need to be sent across the air interface is because it is a broadcast medium, and the NO needs to have some means of distinguishing between users, and users need to have some way of deciding which communications are intended for them.

If symmetric encipherment is being used, then it is impossible to encipher the addresses. This is because the NO needs to know which key to use to decipher an address, i.e. the NO needs to read the address *before* deciphering it. Similarly, a user needs to read an address embedded in an enciphered data string before deciding whether it should attempt to decipher it. Of course, these problems disappear if all the entities use the same key, but this is very insecure and we do not consider this approach further here.

This has led to the use of temporary identities with symmetric encipherment (as in GSM) where the RI is not used as an address, and instead a 'temporary' address is used to identify a user, and this temporary address changes at regular intervals. The new temporary address is chosen by the NO and is sent to the user in enciphered form, hence preventing an interceptor from linking old temporary addresses to new ones. The problem with the GSM approach of this type is the need to use the user's RI prior to setting up an initial temporary address. However, this problem can be avoided by using two levels of temporary addressing, as in the approach documented in subsection 9.3 of this report.

When asymmetric encipherment is used it is now possible to encipher addresses, at least on the 'up link', i.e. in communications between mobile users and an NO. This is because all users can encipher the data they send to the NO using the NO's public encipherment key. Protecting the 'down link' is rather more problematic however, and will still require the use of some form of temporary address. However the 'set up' problems associated with GSM can probably be avoided by using this approach. The only remaining problem is to ensure that a user:

a) knows which NO it is sending to (and hence can use the right public key), and

b) possesses reliable copies of public encipherment keys for all NOs it may wish to use for mobile communications.

One disadvantage of the use of asymmetric encipherment techniques is that they are typically much more computationally complex than symmetric encipherment techniques.

An example of an ILP and entity authentication scheme based on asymmetric encipherment is given in subsection 9.2.5 below. A further example can be found in subsection 9.4. In addition Beller et al., [101], have proposed a privacy and authentication scheme using asymmetric cryptographic techniques on a portable communications system.

9.2.4.3 Legal and operational limitations on ILP

In our discussion of ILP requirements in subsection 9.2.4.1, we have ignored the legal requirements which may apply to NOs. There are two issues which may affect the provision of ILP.

- The *Calling Line Identifier (CLI)* requirement, applying in some management domains, that called entities are provided with the CLI (which typically means the telephone number) of the party calling them.

- The requirement, again applying in some management domains, for law enforcement and/or security agencies to be given access to certain calls starting or terminating within its domain. Depending on the legal framework in operation, such access may only be granted when certain legal niceties have been completed (e.g. the issuing of an interception warrant). This interception requirement could potentially be applied to all calls routed through a domain, albeit that they do not start or terminate within that domain, although this is a somewhat controversial area. For further details of the evolving European rules in this area see [102].

Ultimately this means that some NOs may need to know the RI of users sending and/or receiving calls within their network. However, this still does not mean that it is a good idea for the ILP scheme used to necessitate the transfer of a user's RI to a NO. It would be far more appropriate to have RIs routinely transferred from SPs to NOs only when NOs need them for legal and/or operational reasons.

Thus one could envisage a situation where some NOs will (by law) not provide service to a user unless the user's SP is prepared to provide the user's RI to the NO. In addition, some users may be so concerned about privacy that they refuse to use their mobile telephone in networks where their RI has to be divulged. Hence, if the ILP mechanism can avoid the need for the user's RI to be distributed outside the SP, a whole range of privacy options become possible, giving both users and government agencies the maximum flexibility to manage identity and location privacy.

9.2.5 An ILP mechanism based on asymmetric encipherment

We now present an example of an ILP mechanism for 3GS based on the use of asymmetric encipherment. The four roles involved in this mechanism: Users, NOs, SPs and Intruders, are defined in Clause 3.2 of Technical Report 1, [3].

9.2.5.1 Requirements

The protocol we present is actually based on a combination of public key encipherment and symmetric cryptographic techniques. Nonces are used for checking timeliness.

The following cryptographic functions are used.

- A public key encipherment function (of which the user and SP need to possess implementations). We use $\{X\}_{K+}$ to denote the public key encipherment of data X using the public encipherment key $K+$.

- A cryptographic check function f (of which the user and SP need to possess implementations). We use $f_K(X)$ to denote the output of f given input data X and key K.

- A symmetric encipherment function e (of which the user, NO and SP need to possess implementations). We use $e_K(X)$ to denote the output of e given input data X and key K. Note that this encipherment algorithm needs to provide integrity and origin authentication properties (c.f. requirements (a) and (b) in Clause 4 of ISO/IEC 11770-2, [5]). Note also that, if necessary, the encipherment algorithms used by the three pairs: user/NO, user/SP, and NO/SP, can all be distinct; we have assumed that a single algorithm is used to simplify the presentation.

The following keys need to be in place:

- The SP needs to generate a public key/private key pair for the public key encipherment algorithm. The user needs to be provided with a reliable copy of the SP's public encipherment key (which we denote K_{S+}).

- The user and SP need to share a secret key for the cryptographic check function f (which we denote K'_{US}).

- The two entity pairs: user/SP, and NO/SP, both need to share a secret key for the symmetric encipherment algorithm, which we denote by K_{US} and K_{NS} respectively.

In addition the user, NO and SP all need to have the means to generate non-repeating nonces, and the SP also needs to have the means to generate temporary identities and session keys.

9.2.5.2 The protocol

The following protocol (almost) conforms to Key Establishment Mechanism 9, specified in Clause 6.3 of ISO/IEC ISO/IEC 11770-2, [5]. The only point at which it diverges from the standard is that the user's identity *U* is never sent in clear text and is known only to the SP and itself (the standard protocol would require *U* to be sent in clear text in message M2). The NO and SP identities, *N* and *S* respectively, can be transmitted in clear text.

In the protocol description $U \rightarrow N$: *m* means that *U* sends message *m* to *N*, and $X||Y$ denotes the concatenation of data items *X* and *Y* in the order specified.

M1: $U \rightarrow N$: $R_U||S||N||\{R_U||U||N||f_{K'_{US}}(R_U||U||N)\}_{K_{S+}}$

M2: $N \rightarrow S$: $R_N||R_U||S||N||\{R_U||U||N||f_{K'_{US}}(R_U||U||N)\}_{K_{S+}}$

M3: $S \rightarrow N$: $e_{K_{NS}}(R_N||K_{UN}||T_U)||e_{K_{US}}(R_U||K_{UN}||N||T_U)$

M4: $N \rightarrow U$: $e_{K_{US}}(R_U||K_{UN}||N||T_U)||e_{K_{UN}}(R'_N||R_U||T_U)$

M5: $U \rightarrow N$: $e_{K_{UN}}(R_U||R'_N||N)$

The values R_U, R_N and R'_N are nonces generated by *U*, *N* and *N* respectively. K_{UN} is a session key for *U* and *N*. T_U is a new temporary identity for *U*, to be used by the user and NO.

9.2.5.3 Procedure

The procedure associated with this protocol is as follows. Note that if, at any point, a check fails then the protocol is aborted.

1. The user generates and stores a nonce R_U. The user then sends the NO an authentication request **M1** containing R_U, in which it lets the NO know its SP is *S*. The user's real identity (*U*) is enciphered using K_{S+} so that only the SP can read it.

2. In **M2** the NO forwards the user's request to the SP. The NO also appends (and stores) a nonce R_N.

3. On receipt of **M2** the SP first deciphers the enciphered string using its private decipherment key. The SP then checks the output of *f* using its copy of K'_{US}. The SP assigns a temporary user identity T_U and a session key K_{UN} for use by *U* and *N*, and distributes them to the NO and user in **M3**. The SP maintains a database of relationships between users *U* and temporary identities T_U.

4. On receipt of **M3**, the NO first deciphers (and simultaneously integrity checks) the first part of the message. The NO then checks that the nonce it contains is correct, and also uses the nonce to link the message with the correct 'transaction'. The NO then retrieves the new temporary identity T_U and session key K_{UN}, and uses the latter to generate the second part of message **M4** which contains a second nonce, R'_N, which the NO also stores.

5. On receipt of **M4**, the user first deciphers (and simultaneously integrity checks) the first part of the message. The user checks that the nonce it contains is correct, and retrieves the new session key K_{UN} and temporary identity T_U. The session key is then used to decipher (and simultaneously integrity check) the second part of the message, and the user now checks that the nonce it contains is correct. Finally the user uses the session key to generate message **M5**.

6. On receipt of **M5**, the NO first deciphers (and simultaneously integrity checks) it, and then checks the nonce R'_N.

In this protocol the NO is not given the user's RI, and can only identify a user by the temporary identity T_U supplied by the SP. The NO will use the temporary identity T_U when communicating with SP in order to be recompensed for the cost of providing service to the user.

9.2.5.4 Performance

This protocol can be used to provide service related authentication option R1 defined in subsection 5.2, in which the authentication information about the user is provided by its SP.

This protocol has the following advantages as compared with the GSM ILP approach mentioned in subsection 9.2.2.

1. User RIs are never transmitted in clear text either in the mobile radio path or in the fixed channel between NO and SP.
2. NOs are not given access to a user's RI (unless that is necessary for legal or other non-technical reasons).
3. Authentication of both NO and SP is implicitly included.

This protocol can prevent threats T1, T2, T3, T4, T5 and T6 listed in subsection 9.2.3. Threat T7, i.e. that an SP abuses user identity and location information, can only be prevented by internal management controls imposed by an SP.

The cost of this protocol as compared with conventional protocols, for example those presented in subsection 5.3, is as follows:

1. each SP must have a public key known to its all users and keep a corresponding private key secret, and
2. each user has to compute $\{R_U||U||N||f_{K'_{US}}(R_U||U||N)\}_{K_{S+}}$, which has then to be checked by the SP.

9.2.6 Conclusions

This report discusses ILP mechanisms and proposes two fundamental approaches to the provision of ILP and entity authentication in 3GS. An example of a scheme based on asymmetric cryptography has been provided. The advantages of the example protocol are as follows:

1. the user RI is never transmitted in clear text, either in the mobile radio path or in the fixed channel between NO and SP,
2. NOs need never know the user RI (unless legally required to do so),
3. session key distribution between a user and NO is included, and
4. authentication among users, NOs and SPs is provided.

9.3 A combined ILP and authentication mechanism using symmetric cryptography

The text in this section is partly based on two papers, [103,104], prepared for publication as part of the 3GS3 project.

In GSM networks it is theoretically possible for an intruder to masquerade as a network operator by imitating a base station, as GSM only provides *unilateral* authentication of a user to a network operator. In the case of GSM it is difficult to see how the intruder could obtain much benefit from doing this. However, in third-generation systems it is likely that network operators will have considerably more over-the-air control of users. For instance, they may be able to disable faulty terminals directly, or write billing data direct to the UIM (the UMTS equivalent of a SIM). For this reason a *mutual* (two-way) entity authentication mechanism is necessary.

The mechanism we describe here is based on the use of secret key cryptography and provides mutual entity authentication between the user and the network operator.

9.3.1 Advantages

It sets up (and uses) a temporary key between a user and a network operator. This means that there is no need for communication between a network operator and a service provider once a user has successfully registered with a network operator. This is in contrast with the existing GSM mechanism, which requires regular communications between network operator and service provider to transfer challenge-response pairs.

It also combines the provision of user identity confidentiality, entity authentication and session key generation in a single mechanism.

The mechanism also conforms to the relevant ISO/IEC standard, [4].

9.3.2 Possible restrictions on user identity confidentiality

Note that, in the mechanism described here, the network operator is not automatically given the user's *IMUI* (International Mobile User Identity). If this is necessary for legal and/or operational reasons it can be included in the third message of the 'new registration' authentication mechanism.

9.3.3 Security features provided by the mechanism

The mechanism provides the following security features:

1. Mutual entity authentication between the user and the network operator.
2. User identity confidentiality over the communications path between the user and the network operator.
3. Session key establishment between the user and the network operator for use in providing other security features, possibly including confidentiality and/or integrity for data passed between the user and network operator.

The mechanism makes use of the following types of cryptographic key:

- *user - service provider key* K_{SU}. These are secret keys known only to a user and their service provider. These secret keys remain fixed for long periods of time.
- *user - network operator key* K_{NU}. These are secret keys known only to a user and their 'current' network operator. These keys may remain fixed while a user is registered with a particular network operator. Associated with every such key is a Key Offset (*KO*), which is used in conjunction with the user - service provider key K_{SU} to generate K_{NU}.
- *session key KS*. These are secret keys also known only to the user and their current network operator (i.e. the network operator with whom they are registered). A new session key is generated

as a result of every use of the authentication mechanism. These keys can be used for data encipherment, and/or for the provision of other security features.

The mechanism makes use of the following cryptographic algorithms:

- *user authentication algorithm* A_U. This algorithm takes as input a secret key and a data string and outputs a check value *RES*.
- *service provider authentication algorithm* A_S This algorithm takes as input a secret key and a data string and outputs a check value *RES*. This algorithm may be the same as or distinct from the algorithm A_U.
- *identity hiding algorithm* C_U. This algorithm takes as input a secret key and a data string and outputs a string *CIPH* used to conceal a user identity.
- *session key generation algorithm* A_K. This algorithm takes as input a secret key and a data string and outputs a session key K_S.
- *user - network operator key generation algorithm* A_N. This algorithm takes as input a secret key and a data string and outputs a user - network operator secret key K_{NU}. This algorithm may be the same as or distinct from the algorithm A_K.

The mechanism makes use of the following types of identifiers:

- *International Mobile User Identity IMUI*. This is an identity permanently associated with a user. The IMUI is never passed across the air interface, thus preventing its unauthorised disclosure.
- *network operator identity NOID*.
- *temporary user identity for network operator* $TMUI_N$. This (temporary) identity is used to identify a user to the network operator with which they are currently registered. It is known to the user and to the current FPLMTS network operator.
- *temporary user identity for service provider* $TMUI_S$. This (temporary) identity is used to identify a user to its service provider. It is known to the user and to its service provider.

9.3.4 The mechanism

There are two versions of the mechanism, depending on whether or not the user is currently registered with the network operator. We consider the two cases separately (although they are closely related).

9.3.4.1 Current registrations

We first consider the case where the user is already registered with the network operator. This means that the user and the network operator will share a valid temporary identity $TMUI_N$ and secret key K_{NU}. The mechanism for this case consists of three messages exchanged between the user and the network operator. The service provider is not involved.

The three messages are as follows.

1. **user** $\rightarrow$ **NO**: $TMUI_N$, RND_U
2. **NO** $\rightarrow$ **user**: RND_N, $TMUI'_N \oplus CIPH_N$, RES_N
3. **user** $\rightarrow$ **NO**: RES_U

The values RND_U and RND_N are random 'challenges' generated by the user and the network operator respectively.

The values RES_U and RES_N are 'challenge responses' generated by the user and the network operator respectively. RES_N is calculated using the user authentication algorithm A_U with key input K_{NU} and data string input the concatenation of RND_N, RND_U and $TMUI'_N$. RES_U is calculated using the user

authentication algorithm A_U with key input K_{NU} and data string input the concatenation of RND_U and RND_N.

$TMUI'_N$ is the 'new' temporary user identity for use with the network operator. This will replace the current temporary identity $TMUI_N$

$CIPH_N$ is a string of bits used to conceal the new temporary identity $TMUI'_N$ whilst it is in transit between the network operator and the user. It is calculated using the identity hiding algorithm C_U with secret key input K_{NU} and data string input RND_U.

The user and the network operator can compute a session key K_S as the output of the session key generation algorithm A_K when given secret key input K_{NU} and data string input the concatenation of RND_U, RND_N and $TMUI'_N$.

9.3.4.2 New registrations

We second consider the case where the user is not registered with the network operator. This means that the user and the network operator do not share any information. The mechanism for this case consists of five messages exchanged between the user, the network operator, and the service provider of the user.

The five messages are as follows.

1. **user** $\rightarrow$ **NO**: $TMUI_S$, RND_U
2. **NO** $\rightarrow$ **SP**: $TMUI_S$, RND_U
3. **SP** $\rightarrow$ **NO**: $TMUI'_S \oplus CIPH_S$, KO, K_{NU}, RES_S
4. **NO** $\rightarrow$ **user**: $TMUI'_S \oplus CIPH_S$, KO, RES_S, RND_N, $TMUI'_N \oplus CIPH_N$, RES_N
5. **user** $\rightarrow$ **NO**: RES_U

First note that we assume that a secure channel is available for exchanging messages 2 and 3 between the network operator and service provider.

As previously, the values RND_U and RND_N are random 'challenges' generated by the user and the network operator respectively.

The values RES_U, RES_N, and RES_S are 'challenge responses' generated by the user, network operator, and service provider respectively. RES_N and RES_U are calculated as in the previous case. RES_S is calculated using the service provider authentication algorithm A_S with key input K_{SU} and data string input the concatenation of RND_U, KO and $TMUI'_S$.

$TMUI'_S$ is the 'new' temporary user identity for use with the service provider. This will replace the current temporary identity $TMUI_S$. As previously, $TMUI'_N$ is the 'new' temporary user identity for use with the network operator.

$CIPH_S$ is a string of bits used to conceal the new temporary identity $TMUI'_S$ whilst it is in transit between the service provider and the user. It is calculated using the identity hiding algorithm C_U with secret key input K_{SU} and data string input RND_U. As previously, $CIPH_N$ is a string of bits used to conceal the new temporary identity $TMUI'_N$ whilst it is in transit between the network operator and the user.

On receipt of message 4, the user can compute the network operator secret key K_{NU} as the output of the network operator key generation algorithm A_N, when given as secret key input K_{SU}, and data string input the key offset KO concatenated with the network operator identity $NOID$ (this same calculation is done by the service provider on receipt of message 2).

As previously, the user and the network operator can compute a session key K_S as the output of the session key generation algorithm A_K when given secret key input K_{NU} and data string input the concatenation of RND_U, RND_N and $TMUI'_N$.

Note that, as a result of the above mechanism, the user and the network operator will share a secret key K_{NU} and a temporary identity $TMUI'_N$.

9.3.5 Conclusion

The proposed mechanism provides several security features that will be required for third-generation systems. Further work is required to establish the efficiency of the mechanism, and to determine how the mechanism can be managed. Other work within the 3GS3 project has involved using a variant of the SVO logic, [105], to verify the correctness of this mechanism. In fact, analysis of the protocol using this logic revealed a subtle flaw in an earlier version of the protocol which has now been corrected. For further details of one such formal analysis see Section 9.5.

9.4 A combined ILP and authentication mechanism using asymmetric cryptography

9.4.1 Introduction

In this subsection we propose a family of three mutual authentication mechanisms for UMTS, which are based on the use of asymmetric cryptographic techniques.

The entities involved in the exchanges of messages are a mobile user, a network operator (NO), and a service provider (SP), as defined in subsection 5.2.2.1. Two of the three mechanisms also rely on the existence of a *Certification Authority (CA)* to issue signed certificates for the user's and NO's public encipherment keys. All entities are assumed to have the means to verify these certificates. The CA need not be on-line during use of the mechanisms.

The mechanisms provide the following security services.

- Mutual entity authentication between the user and NO.
- User identity confidentiality over the mobile radio path between the user and NO and in the fixed channel between the NO and SP.
- Session key K_{UN} establishment between the user and NO.

The mechanism makes use of the following types of asymmetric encipherment system.

- The user asymmetric encipherment system, including public encipherment transformation E_U and private decipherment transformation D_U. This system is likely to be fixed for long periods of time. The SP is authorised to issue the public encipherment key e_U to the NO.
- The NO asymmetric encipherment system, including public encipherment transformation E_N and private decipherment transformation D_N. This system is also likely to be fixed for long periods of time. The SP is authorised to issue the public encipherment key e_N to the user.
- The SP asymmetric encipherment system, including public encipherment transformation E_S and private decipherment transformation D_S. This system is yet again likely to be fixed for long periods of time. We assume that the user has access to an authenticated copy of the public encipherment transformation E_S.

The mechanism also makes use of a one-way hash-function h.

As throughout, in the mechanism descriptions, $A \rightarrow B$: m indicates that A sends message m to B; $X||Y$ denotes the concatenation of X and Y; and **M*i*** is the ith exchanged message.

9.4.2 The mechanisms

There are three versions of the mechanism, depending on whether or not the user is currently registered with the NO, i.e. whether the user and NO already know the public encipherment transformations of each other before the authentication processing starts, and whether the authentication server is on-line or off-line.

9.4.2.1 Current registrations

The first version of the mechanism applies in the case where the user is already registered with the NO. This means that the user and NO must satisfy the following requirements.

1. The user and NO have their own personal asymmetric encipherment systems (E_U, D_U and E_N, D_N).
2. The user and NO have access to authenticated copies of the public encipherment transformation of one another.
3. The user's and NO's identifiers U and N are known to both of them.

The message exchanges of this version are as follows.

M1: $U \rightarrow N: \; E_N(U||R_U||T_U)$

M2: $N \rightarrow U: \; T_U||E_U(R_N||R_U||N)$

M3: $U \rightarrow N: \; h(R_U||R_N||U)$

The procedural aspects of this protocol are as follows:

1. *U* chooses and stores an unpredictable nonce R_U. *U* also chooses and stores a temporary identity for itself, which we denote by T_U. *U* then uses *N*'s public encipherment transformation to encipher a block made up of its identifier *U*, the nonce R_U, and the temporary identity T_U and sends it to *N*.

2. After receiving **M1**, *N* deciphers it using D_N, chooses and stores another unpredictable nonce R_N, and enciphers a block made up of R_N, R_U and its identifier *N* using E_U. *N* then sends the block back to *U*, if necessary using T_U as the address for *U*.

3. On receipt of **M2**, *U* deciphers it using D_U, and checks the distinguishing identifiers *N* and that the received value R_U agrees with the challenge sent in **M1**. If all checks are successful, *U* accepts the value R_N, constructs the enciphered message **M3** using E_N, and sends it to *N*.

4. On receipt of **M3**, *N* recomputes the hash value using its stored values if R_U, R_N and *U*. If the check is successful, *N* accepts the value R_U.

5. A shared session key K_{UN} can be computed on both sides by combining the two nonces R_U and R_N. The combination function could be a pre-determined one-way hash function. In order to prevent *N* imposing the session key, the choice of this function should be limited; for example, an exclusive-or operation would be a poor choice.

9.4.2.2 New registrations using an off-line Certification Authority

The second version of the mechanism applies in the case where the user is not registered with the NO, and the user and NO do not have any authentication information for each other. However we do assume that the user and NO have valid certificates for their own public encipherment keys.

The following requirements must be satisfied.

1. The user and NO have their own personal asymmetric encipherment systems (E_U,D_U and E_N,D_N).

2. The user and NO have certificates for their own public encipherment keys (Cert_U and Cert_N).

3. The user and NO have access to a trusted copy of the CA's public verification key (so that they can verify each others' certificates).

The exchange messages for this version of the mechanism are as follows.

M1: $N \rightarrow U: \; \text{Cert}_N$

M2: $U \rightarrow N: \; E_N(\text{Cert}_U||R_U||T_U)$

M3: $N \rightarrow U: \; T_U||E_U(R_N||R_U||N)$

M4: $U \rightarrow N: \; h(R_U||R_N||U)$

The procedural aspects of this protocol are as follows:

1. *N* sends *U* **M1** to let *U* know its public encipherment key.

2. After receiving and verifying Cert_N, and thus obtaining a trusted copy of *N*'s public encipherment key, *U* chooses and stores an unpredictable nonce R_U and a temporary identity T_U. *U* then uses *N*'s public encipherment transformation to encipher a block made up of *U*'s own certificate and the nonce and temporary identity it just chose.

3. After receiving **M2**, *N* deciphers it using D_N, retrieves and stores *U*'s nonce and temporary identity, and verifies *U*'s certificate (thereby obtaining a trusted copy of *U*'s public encipherment key). *N* then chooses another unpredictable nonce R_N, and uses *U*'s public encipherment transformation to encipher a block made up of *N*'s newly chosen nonce, together with *U*'s nonce (sent in **M2**) and *N*'s identifier. If necessary *N* uses the temporary identity T_U to address this message.

4. On receipt of **M3**, *U* deciphers it using D_U. *U* now checks that the received value R_U agrees with the challenge sent in **M2**, and that the identifier *N* is the expected value. If the checks are successful, *U* accepts **M3** and the value R_N. *U* then uses the hash-function *h* to construct **M4.**

5. On receipt of **M4**, *N* verifies it using stored values of R_U, R_N and *U*. If the check is successful, *N* accepts the value R_U.

6. As described in the previous version, a shared session key K_{UN} can be computed by combining the two nonces R_U and R_N on both sides. The combination function is a pre-determined one-way hash function. The choice of this function has to prevent *U* imposing the session key.

9.4.2.3 New registrations using an on-line authentication server

The third version of the mechanism applies in the case where the user is not registered with the NO, and the user and NO do not have any authentication information for each other. The SP is involved here as an on-line authentication server. The user and NO must trust the SP to follow the protocol specifications correctly.

The following requirements must be satisfied.

1. The user, NO and SP have their own personal asymmetric encipherment systems (E_U,D_U, E_N,D_N and E_S,D_S, respectively).

2. The user has access to an authenticated copy of the SP's public encipherment transformation E_S.

3. The SP has access to certificates for the user's and NO's public encipherment keys (Cert_U and Cert_N).

4. The user's, NO's and SP's identifiers *U*, *N* and *S* are known to each other.

The exchanged messages for this version of the mechanism are as follows.

M1: $U \rightarrow N: \; S \| E_S(U \| R_U \| T_U)$

M2: $N \rightarrow S: \; N \| E_S(U \| R_U \| T_U)$

M3: $S \rightarrow N: \; E_N(R_U \| \text{Cert}_U \| T_U) \| \text{Cert}_N$

M4: $N \rightarrow U: \; T_U \| E_U(R_N \| R_U \| \text{Cert}_N)$

M5: $U \rightarrow N: \; h(R_U \| R_N \| U)$

The procedural aspects of this protocol are as follows:

1. *U* sends *N* the message **M1** to let *N* know its SP is *S* and to ask *N* to pass his authentication request to *S*. A block made up of the user's identifier, a newly chosen nonce R_U, and a temporary identity T_U, is enciphered using E_S.

2. In **M2**, *N* forwards the user's request with its identifier *N* to *S*.

3. After receiving **M2**, *S* deciphers it using D_S, retrieves *U*'s and *N*'s certificates (Cert_U and Cert_N), and then distributes both these certificates in **M3**.

4. On receipt of **M3**, *N* deciphers the first part of it using D_N to obtain *U*'s certificate, nonce and temporary identity. *N* next verifies the certificate to obtain a trusted copy of *U*'s public encipherment key. *N* then enciphers a block made up of R_N, R_U and Cert_N using U's public

encipherment transformation E_U, and sends this enciphered block as **M4** to U (if necessary using T_U as an address).

5. On receipt of **M4**, U deciphers it using D_U. U checks that the received value R_U agrees with the challenge sent in **M1**. If the check is successful, U accepts **M4** and the nonce R_N. U then verifies the certificate to obtain a trusted copy of N's public encipherment key. U now hashes a block made up of the two nonces and the identifier of U and sends it as M5 to N.

6. On receipt of **M5**, N verifies it using stored values of the two nonces and U's identifier. If the check is successful, N accepts the value R_N.

7. As described in the previous version, a shared session key K_{UN} can be computed by combining the two nonces R'_U and R_N on both sides. The combination function is a pre-determined one-way hash function. The choice of this function has to prevent U imposing the session key.

9.4.3 Properties of the mechanism

The above mechanism has the following properties.

1. The mechanism provides explicit mutual entity authentication of user and NO.

2. This mechanism combines the provision of user identity confidentiality, mutual entity authentication and session key distribution in a single mechanism (user identity confidentiality being provided by the provision of temporary user identities).

3. The mechanism provides mutual key confirmation between N and U for both candidate keys R_N and R_U.

4. This mechanism achieves mutual key control by combining the two candidate keys R_U and R_N on both sides to form a shared secret key K_{UN}. However, if necessary, the combination function must be chosen correctly to prevent either entity from choosing the shared secret key.

9.4.4 Further considerations

We now consider some additional requirements which may affect the design of the mechanism.

1. If the NO and SP have a secure channel which is available for exchanging messages 2 and 3 in Version 3 of the mechanism, the first block of message 3 does not have to be enciphered using E_N, so that the message will be

$$\textbf{M3}:\ S \rightarrow N:\ R_U \,||\, \mathrm{Cert}_U \,||\, T_U \,||\, \mathrm{Cert}_N$$

2. The last message in each of the three versions of the mechanism is solely used for completing key confirmation and mutual entity authentication. If these two services are not required then the last message may be omitted in each case.

3. In all three versions of the mechanism, a key transport mechanism is used to form a shared session key K_{UN}. If required, a key agreement mechanism can be used instead. The exchanged messages are the same as above. For example, in Version 1, U chooses an unpredictable number r_u in a set H, computes a function $F(r_u,g)$, keeps r_u secret, and uses $F(r_u,g)$ instead of R_U; similarly, N chooses an unpredictable number r_n in H, computes a function $F(r_n,g)$, keeps r_n secret, and uses $F(r_n,g)$ instead of R_N. After receiving $F(r_n,g)$ and $F(r_u,g)$, U and N proceed respectively to compute the shared key as $K_{UN} = F(r_u,F(r_n,g))$ and $K_{UN} = F(r_n,F(r_u,g))$. Details of the properties which H, g and F must possess are given in ISO/IEC 11770-3, [5].

4. There may be a requirement in UMTS, as discussed in subsection 9.2, for the user real identity U to only be known to the user and the SP, not to the NO. In such a case, not only a temporary user identity but also a temporary user public encipherment transformation would have to been used.

9.5 Evaluation of a combined ILP/authentication mechanism

A detailed evaluation of the mutual authentication mechanism proposed in subsection 9.4 has been performed as part of the process of submitting this mechanism to the UMTS and FPLMTS standardisation processes. A detailed discussion of this work can be found in [106]. This technical report, prepared as part of the 3GS3 project, is too long and specialised to include here. However, copies can be obtained by writing to: *Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England* (quoting the title and number of the report).

10. Formal analysis of mechanisms

Within the 3GS3 project considerable effort has been put into considering how formal (logical) techniques can be applied to the analysis of security mechanisms. Our motivation has been to try and use all the means available to us to test the validity and security of mechanisms proposed for use in UMTS, FPLMTS and other future mobile telecommunications systems. Work has included the use of a variant of the SVO logic to analyse the ILP and mutual authentication mechanisms discussed in subsection 9.3.

A detailed discussion of this work can be found in [107]. This technical report, prepared as part of the 3GS3 project, is too long and specialised to include here. However, copies can be obtained by writing to: *Computer Science Department, Royal Holloway, University of London, Egham, Surrey TW20 0EX, England* (quoting the title and number of the report).

11. Areas for further research

11.1 Introduction

As has been discussed at various points in this report, although the 3GS3 project has made a number of successful contributions to the development of security features in the UMTS and FPLMTS standards, further research is still needed in a number of areas. In this brief section we attempt to summarise some of the most significant of these.

11.2 Selecting authentication mechanisms

In subsection 4.2 we reviewed the main options for the type of cryptographic mechanism which might be used to support the provision of mobile user authentication in 3GS. This review raises a number of important points which need to be carefully considered if the most appropriate choice for 3GS security mechanisms is to be made.

- How important is it to provide mechanisms which do not require operators to request user dependent data from the service providers?
- What are the likely political ramifications of standardising on one algorithm in preference to another - especially if the scheme is to be asymmetric?
- What is an acceptable figure for the amount of data which needs to be transferred on the radio path, and what is the trade-off between this and the amount of user data which needs to be signalled between operators and service providers?
- The above points raise the whole question of the balance between the amount of data carried by a user in its access device and the amount which an operator needs to request from the service provider when the user appears on its network.
- How important is it for service providers to be able to choose their own algorithms?
- What are the likely capabilities of access devices - will smart cards which are able to support quite complex asymmetric schemes be available in the year 2000 - will the adoption of such schemes for 3GS help accelerate such capability?
- User authentication is only one security feature required in 3GS. When selecting the mechanism, consideration needs to be given to the other features so that re-use of the mechanism to provide or support other features is maximised. We do not want 20 entirely different mechanisms for 20 features, if this can be avoided.
- Consideration needs to be given to the security requirements at various interfaces for the different options for security mechanisms. For example, with symmetric schemes there is usually a need to provide both confidentiality and authentication for security related data passed between service providers and operators. For asymmetric schemes the requirement is usually for authentication alone. Moreover, with asymmetric schemes, the data is transferred infrequently - perhaps just once.

In subsection 4.4 we considered how authentication can be achieved in a context where a multiplicity of authentication servers are used, not all of which are necessarily trustworthy. Possible future topics for research in this area include the following.

- Can efficient protocols be designed for the case where $\leq n/2$ of the authentication servers are trustworthy? In subsection 4.5 we showed that, in some cases, the answer to this question is 'Yes'.
- As is common in the design and analysis of distributed protocols, it would be useful to distinguish between failed authentication servers, which fail to take part in protocols, and malicious authentication servers, which participate in a dishonest way. Protocols could then be designed to deal with various proportions of failed and dishonest servers. Again this issue has partly been addressed in subsection 4.5.
- The derivation of lower bounds on the number of messages in various types of protocol would give a measure on how efficient specific protocols are.
- It would be of interest to see if more efficient protocols could be designed based on the use of timestamps and synchronised clocks as opposed to the use of nonces. Typically, timestamp-based protocols require fewer messages than nonce-based protocols.

In subsection 4.5 we continued the discussion from subsection 4.4, and considered the case where only a minority of the servers may be trustworthy. The protocol presented in subsection 4.5 does not meet likely legal requirements for warranted interception (c.f. subsection 7.2), because only the two users

know the resultant session key if the protocol is successful. As we have discussed, in some cases, governments have requirements to intercept user traffic in order to combat crime and protect national security, see e.g. [75]. So a possible topic for future research in this area is how to provide an authentication and key distribution service in which no untrustworthy server can compromise the users' secrets, as well as meeting the legal requirements for warranted interception in the domains it serves.

11.3 Service-related authentication

In section 5 of this report we considered a number of options for the provision of service-related authentication in 3GS. These options were presented within the context of two models for 3GS, a role model and a functional model. Examples of protocols fitting various model options and falling within various defined categories were given. It was noted in subsection 5.3.1 that no protocols in category P4 are considered, i.e. protocols based on zero knowledge techniques, although this appears to be a topic worthy of future study.

Based upon the discussion and analysis in section 5, we suggest that future research on service-related authentication and session key distribution in 3GS should include the following.

- The investigation of the correctness of each of the protocols which have been discussed in section 5 of this report, and any others likely to be of relevance to 3GS, applying the logics of Burrows-Abadi-Needham, [108], (and derivatives, [109], [110], [111]) and Meadows-Syverson, [112], [113], [114], [115], [116].
- The study of the key management problem, in particular shared secret key distribution between the NOs and SPs.
- The study of how session key distribution might be achieved using protocols of types P2 and P3.
- The analysis of terminal equipment and the corresponding roles, and to consider if they should be involved in service related authentication in 3GS.
- The development and testing of new authentication and session key distribution protocols.
- The identification of any shortcomings of the existing ISO/IEC authentication protocol standard (ISO/IEC 9798, [4]) with respect to the needs of 3GS. This may lead us to specifying requirements for additional parts of ISO/IEC 9798.
- The investigation of the trust relationships implicit in the various role model authentication options, and investigations of the design of authentication protocols appropriate to the levels of trust which are likely to exist between the entities involved (see also subsection 5.2.5).

It is certainly the case that the development and testing of authentication and session key distribution protocols remains a research topic of considerable importance.

11.4 Encryption and multiple access

In section 6 we considered the impact of the multiple access method employed in the air interface on the way in which data sent across the air interface might be encrypted. In subsection 6.3 we considered the special case of TDMA systems. The interaction between encryption and two possible TDMA enhancements (dynamic channel allocation and slow frequency hopping) were noted as topics worthy of future study.

11.5 Key management for end-to-end security

In section 7 we considered the problem of establishing keys for use in providing end-to-end security services. In subsection 7.3 we considered one way in which multiple Trusted Third Parties, not all of whom may be trustworthy, might be used to simultaneously provide key distribution and a key escrow service. The following open questions regarding this scheme are of potential practical importance.

- Do there exist practical key escrow systems forcing users to use only the current escrowed session key?
- Can a practical key escrow scheme be designed for the case where more than two domains are involved, and where escrow agencies are not permitted to span more than one domain?

A rather different approach to the problem of distributing keys to support end-to-end security services was considered in subsection 7.4, based on the 'wiretap channel' concepts of Wyner. The schemes described there all appear to be theoretically sound; however it remains to be seen though, whether practical implementations will be forthcoming. An obvious direction for future work is to investigate such possibilities.

12. Bibliography

[1] LINK PCP, 3GS3, Technical Report 1: *Security features for third generation systems*, Version 1, 4th February 1994.

[2] LINK PCP, 3GS3, Technical Report 1: *Security features for third generation systems*, Version 2, 1st May 1995.

[3] LINK PCP, 3GS3, Technical Report 1: *Security features for third generation systems*, Version 3 (final version), 14th February 1996.

[4] ISO/IEC 9798, Parts 1-5, *Information technology - Security techniques - Entity authentication*, 1991 (Part 1), 1993 (Part 3), 1994 (Part 2), 1995 (Part 4), (Part 5 is as yet unpublished).

[5] ISO/IEC 11770, Parts 1,2,3, *Information technology - Security techniques - Key management.*

[6] ISO 7498-2, *Information processing systems - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture*, February 1989.

[7] ISO/IEC 10181, Parts 1-7, *Information technology - Open Systems Interconnection - Security frameworks for open systems*.

[8] C.J. Mitchell, *Security techniques*. Presented at *IEE Electronics Division Colloquium on Security in Networks (London, February 1995).* **IEE Digest No. 1995/024**, pp.2/1-2/6.

[9] ISO/IEC 9796, *Information technology - Security techniques - Digital signature scheme giving message recovery*. September 1991.

[10] ISO/IEC 14888, Parts 1-3, *Information technology - Security techniques - Digital signatures with appendix*.

[11] ISO/IEC 10116, *Information technology - Modes of operation for an n-bit block cipher algorithm*, September 1991.

[12] ISO/IEC 9979, *Data cryptographic techniques - Procedures for the registration of cryptographic algorithms*, December 1991.

[13] ISO/IEC 9797, *Information technology - Security techniques - Data integrity mechanism using a cryptographic check function employing a block cipher algorithm*, April 1994 (2nd edition).

[14] ISO/IEC 10118, Parts 1-4, *Information technology - Security techniques - Hash-functions*, October 1994 (Parts 1,2).

[15] ISO/IEC 13888, Parts 1-3, *Information technology - Security techniques - Non-repudiation*.

[16] L. Chen, D. Gollmann and C. Mitchell, *Tailoring authentication protocols to match underlying mechanisms*, to be presented at the *Australasian Conference on Information Security and Privacy*, Wollongong, NSW, Australia, June 1996.

[17] L. Gong, *Increasing availability and security of an authentication service. IEEE Journal on Selected Areas in Communications* **11** (1993) 657-662.

[18] R. Yahalom, B. Klein and T. Beth, *Trust-based navigation in distributed systems.* European Institute for System Security, Karlsruhe University, Technical Report **93/4**, 1993.

[19] C. Mitchell, *Security in future mobile networks*. Proceedings of the 2nd International Workshop on Mobile Multi-Media Communications (MoMuC-2), Bristol, April 1995.

[20] C.J. Mitchell and A. Thomas, *Standardising authentication protocols based on public key techniques*. *Journal of Computer Security* **2** (1993) 23-36.

[21] L. Gong, *Variations on the themes of message freshness and replay.* In: *Proceedings: the Computer Security Foundations Workshop VI*, pages 131-136. IEEE Computer Society Press, Los Alamatos, California. 1993.

[22] S.M. Bellovin and M. Merritt, *Limitations of the Kerberos authentication system*. *Computer Communications Review* **20(5)** (October 1990) 119-132.

[23] I. Damgard, *Towards practical public key systems secure against chosen ciphertext attacks.* In: *Advances in Cryptology - CRYPTO '91*, pages 445-456. Springer-Verlag, Berlin, 1991.

[24] K.Y. Lam and T. Beth, *Timely authentication in distributed systems*. In: *Proceedings ESORICS 92*, pages 293-303. Springer-Verlag, Berlin, 1992.

[25] M. Reiter, K. Birman and R. van Renesse, *Fault-tolerant key distribution.* Department of Computer Science, Cornell University, Technical Report **93-1325**, January 1993.

[26] B. Simons, J.L. Welch and N. Lynch, *An overview of clock synchronisation.* In: *Advances in Fault-tolerant distributed computing*. Springer-Verlag, 1990.

[27] K.Y. Lam, *Building an authentication service for distributed systems. Journal of Computer Security* **2** (1993) 73-84.

[28] L. Gong, *A security risk of depending on synchronized clocks. ACM Operating Systems Review* **26(1)** (January 1992) 49-53.

[29] L. Chen, D. Gollmann and C. Mitchell, *Key distribution without individual trusted authentication servers*. In: *Proceedings: the Eighth IEEE Computer Security Foundations Workshop, June 1995*, pages 30-36. IEEE Computer Society Press, Los Alamatos, California. 1995.

[30] L. Chen, D. Gollmann and C. Mitchell, *Distributing trust amongst multiple authentication servers. Journal of Computer Security*, to appear.

[31] E.F. Brickell and D.R. Stinson, *Authentication codes with multiple arbiters*. In: *Advances in Cryptology - Proceedings Eurocrypt '88*, pages 51-54. Springer-Verlag, Berlin, 1988.

[32] B. Klein, M. Otten and T. Beth, *Conference key distribution protocols in distributed systems.* In: *Codes and Cyphers, Proceedings of the fourth IMA Conference on Cryptography and Coding*, pages 225-241. Formara Ltd., Southend-on-Sea, Essex, 1995.

[33] T.P. Pedersen, *A threshold cryptosystem without a trusted party*. In: *Advances in Cryptology - Proceedings Eurocrypt 91*, pages 522-526. Springer-Verlag, Berlin, 1991.

[34] S.C. Kothari, *Generalized linear threshold scheme.* In: *Advances in Cryptology - Proceedings Crypto 84*, pages 231-241. Springer-Verlag, Berlin, 1985.

[35] A. Shamir, *How to share a secret. Communications of the ACM* **22** (1979) 612-613.

[36] L. Chen, G. Gollmann and C.J. Mitchell, *Authentication in distributed systems using minimally trusted servers*. To be submitted.

[37] M. Steiner, G. Tsudik and M. Waidner, *Refinement and extension of encrypted key exchange*. Discussion paper, IBM Zürich Research Laboratory, Rüschlikon, Switzerland, December 1994.

[38] G.J. Simmons and C. Meadows, *The role of trust in information integrity protocols. Journal of Computer Security* **3** (1994/95) 71-84.

[39] J.J. Tardo and K. Alagappan, *SPX: Global authentication using public key certificates.* In: *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 232-244. IEEE Computer Society Press, Los Alamitos, California. May 1991.

[40] ETSI/PT12 GSM-03.20, *Security related network functions.* August 1992.

[41] J.A. Bull, L. Gong and K.R. Sollins, *Towards security in an open systems federation.* In: *Proceedings of ESORICS 92*, Toulouse, France, November 1992.

[42] L. Gong, *Using one-way function for authentication. Computer Communication Review* **19** (5), pp.8-11, October 1989.

[43] G. Tsudik and E.V. Herreweghen, *On simple and secure key distribution.* In: *Proceedings of the 1st ACM Conference on Computer and Communications Security*, November 1993.

[44] D.J. Goodman and S.X. Wei, *Efficiency of packet reservation multiple access. IEEE Transactions on Vehicular Technology* **40** (1991) pp.170-176.

[45] S. Nanda, D.J. Goodman and U. Timor, *Performance of PRMA: A packet voice protocol for cellular systems. IEEE Transactions on Vehicular Technology* **40** (1991) pp.584-598.

[46] J. Dunlop, *Packet access mechanisms for cellular radio*, IEE Electronics and Communication Engineering Journal, June 1993, pp.173-179.

[47] R.S. Mowbray and P.M. Grant, *Wideband coding for uncoordinated multiple access communications*. *IEE Electronics and Communication Engineering Journal*, December 1992, pp.351-361.

[48] D.V. Sarwate and M.B. Pursley, *Cross-correlation properties of pseudo-random and related sequences*. P*roc. IEE* **68** (1980).

[49] R. Steele and J.E.B. Williams, *The intelligent multi-mode mobile portable for third generation networks*. UK CCIR TG8/1 CP151, February 1992.

[50] T. Kasami, *Weight distribution formula for some class of cyclic codes.* Coordinated Science Laboratory, University of Illinois, Urbana, Technical Report No. **R-285**, April 1966.

[51] R. Gold, *Optimal binary sequences for spread spectrum multiplexing*. *IEEE Transactions on Information Theory* **IT-13** (1967) pp.619-621.

[52] R. Gold, *Maximal recursive sequences with 3-valued recursive cross-correlation functions*. *IEEE Transactions on Information Theory* **IT-14** (1968) pp.154-156.

[53] TIA/EIA Interim Standard TIA/EIA/IS-95, *Mobile station - Base station compatibility standard for dual-mode wideband spread spectrum cellular system*. July 1993.

[54] Qualcomm Inc., *An overview of the application of code division multiple access (CDMA) to digital cellular systems and personal cellular networks*. EX60-10010, May 1992.

[55] H. Beker and F. Piper, *Cipher systems: The protection of communications*. Northwood Publications, 1982.

[56] D.J. Torrieri, *Principles of secure communication systems*. Artech House, 1992 (2nd edition).

[57] J.D. Olson, R.A. Scholtz and L.R. Welch, *Bent function sequences*. *IEEE Transactions on Information Theory* **IT-28** (1982) pp.858-864.

[58] S. Prasad and L.C. Quynh, *Class of binary cipher sequences with best possible autocorrelation function*. *IEE Proc. Part F* **132** (1985) pp.576-580.

[59] S. Prasad and L.C. Quynh, *New class of sequence sets with good auto- and crosscorrelation functions*. *IEE Proc. Part F* **133** (1986) pp.281-287.

[60] A. Klapper, A.H. Chan and M. Goresky, *Cascaded GMW sequences*. *IEEE Transactions on Information Theory* **39** (1993) pp.177-183.

[61] A. Klapper, *The vulnerability of geometric sequences based on fields of odd characteristic*. *Journal of Cryptology* **7** (1994) pp. 33-51.

[62] L. Blum, M. Blum and M. Shub, *A simple unpredictable pseudo-random number generator. SIAM Journal on Computing* **15** (1986) pp.364-383.

[63] S. Shepherd, P. Sanders and C. Stockel, *The quadratic residue cipher and some notes on implementation*. *Cryptologia* **17** (1993) pp.262-282.

[64] S. Shepherd and S. Barton, *The use of Blum sequences as secure spreading codes*. *IEE Colloquium on spread spectrum techniques for radio communications systems*, April 1994, pp.8/1-8/6.

[65] J. Brown, *Combined multiple access and encryption for CDMA systems*. In *Proceedings: 3rd International Symposium on Communication Theory and Applications (Ambleside, July 1995).* IEE, 1995.

[66] M.B. Pursley and D.V.Sarwate, *Performance evaluation for phase-coded spread-spectrum multiple-access communication - Part II: Code sequence analysis*. *IEEE Transactions on Communications* **COM-25** (1977) 800-803.

[67] M.B. Pursley and D.V. Sarwate, *Crosscorrelation properties of pseudorandom and related sequences*. *Proc IEEE* **68** (1980) 593-619.

[68] M.B. Pursley, *Performance evaluation for phase-coded spread-spectrum multiple-access communication - Part I: System analysis*. *IEEE Transactions on Communications* **COM-25** (1977) 795-799.

[69] K.H.A. Kärkkäinen, *Mean-Square Cross-Correlation as a Performance Measure for Spreading Code Families*. In *Proc IEEE Second International Symposium on Spread Spectrum Techniques and Applications 1992 (ISSSTA '92)*, pp.147-150.

[70] D.V. Sarwate, *Mean-square correlation of shift register sequences*. *IEE Proceedings part F* **131** (1984).

[71] N. Jefferies, C. Mitchell and M. Walker, *A proposed architecture for trusted third party services*. In *Cryptography: Policy and algorithms - Proceedings: International conference, Brisbane, Australia, July 1995,* Springer-Verlag, Berlin, 1996, pp. 98-104.

[72] N. Jefferies, C. Mitchell and M. Walker, *Trusted third party based key management allowing warranted interception*. In *Proceedings: Public Key Infrastructure Invitational Workshop, MITRE, McLean, VA, USA, September 1985*, **NISTIR 5788**, National Institute of Standards and Technology, Gaithersburg, MD, USA, 1995.

[73] N. Jefferies, C. Mitchell and M. Walker, *Combining TTP-based key management with key escrow*. Submitted to the *Journal of Computer Security*.

[74] D.E. Denning and D.K. Branstad, *A taxonomy for key escrow encryption systems*. *Communications of the ACM* **39 (3)** (1996) 34-40.

[75] National Institute of Standards and Technology, **FIPS Publication 185**: *Escrowed Encryption Standard*, February 1994.

[76] M. Burmester, *On the risk of opening distributed keys.* In: *Advances in Cryptology - CRYPTO '94*, Springer-Verlag (Berlin) (1994) pp. 308-317.

[77] S. Micali, *Fair Cryptosystems.* MIT Technical Report **MIT/LCS/TR-579.b**, November 1993.

[78] U.S. Patent 4956863, *Cryptographic method and apparatus for public key exchange with authentication*. Granted 11th September 1990.

[79] Y. Yacobi, *A key distribution paradox*. In *Advances in Cryptology – CRYPTO '90*, Springer-Verlag, Berlin (1991), pp.268—273.

[80] L. Chen, D. Gollmann and C.J. Mitchell, *A key escrow system in mutually mistrusting domains*. To appear in the *Proceedings of the 4th Cambridge Workshop on Cryptographic Protocols, April 1996*, Springer-Verlag, Berlin.

[81] Y. Frankel and M. Yung, *Escrow encryption systems visited: attacks, analysis and designs.* In: *Advances in Cryptology - CRYPTO '95*, Springer-Verlag, Berlin (1995), pp.222-235.

[82] S. Micali and R. Sidney, *A resilient Clipper-like key escrow system*. MIT Laboratory for Computer Science, November 1994.

[83] J. Kilian and T. Leighton, *Fair cryptosystems, revisited*. In: *Advances in Cryptology - CRYPTO '95*, Springer-Verlag, Berlin (1995), pp.208-221.

[84] S. Micali and R. Sidney, *A simple method for generating and sharing pseudo-random functions, with applications to Clipper-like key escrow systems*. In: *Advances in Cryptology - CRYPTO '95*, Springer-Verlag, Berlin (1995), pp.185-196.

[85] J. Nechvetal, *A public-key based key escrow system*. *Journal of Systems and Software*, to appear.

[86] T.P Pedersen, *Distributed provers with applications to undeniable signatures.* In: *Advances in Cryptology - Eurocrypt '91*, Springer-Verlag, Berlin (1991), pp.221-238.

[87] M.K. Reiter, K.P. Birman and R. van Renesse, *A security architecture for fault-tolerant systems*. *ACM Transactions on Computer Systems* **12** (1994) 340-371.

[88] A.K. Lenstra, P. Winhkler and Y. Yacobi, *A key escrow system with warrant bounds.* In: *Advances in Cryptology - CRYPTO '95*, Springer-Verlag, Berlin (1995) pp.197-207.

[89] U.M. Maurer, *Secret key agreement by public discussion from common information*. *IEEE Transactions on Information Theory* **39** (1993) pp.733-742.

[90] C.E. Shannon, *Communication theory of secrecy systems*. *Bell System Technical Journal* **28** (1949) pp.656-715.

[91] A.D. Wyner, *The wire-tap channel*. *Bell System Technical Journal* **54** (1975) pp.1355-1387.

[92] C.H. Bennet, G. Brassard, C. Crepeau and U.M. Maurer, *Privacy amplification against probabilistic information.* Preprint, 1993.

[93] C.H. Bennet, G. Brassard and J.-M. Robert, *Privacy amplification by public discussion*. *SIAM Journal of Computing* **17** (1988) pp.210-229.

[94] C.H. Bennet, F. Bessette, G. Brassard, L. Savail and J. Smolin, *Experimental quantum cryptography. Journal of Cryptology* **5** (1992) 3-28.

[95] G. Brassard and L. Salvail, *Secret-key reconciliation by public discussion*. In: *Proceedings of Eurocrypt '93*.

[96] C. Mitchell, *A storage complexity based analogue of Maurer key establishment using public channels*. In: *Cryptography and Coding - Proceedings 5th IMA Conference, Cirencester, December 1995*, Springer-Verlag, Berlin (1995), pp.84-93.

[97] U.M. Maurer, *Conditionally-perfect secrecy and a provably-secure randomized cipher*, *Journal of Cryptology* **5** (1992) 53-66.

[98] W. Diffie and M.E. Hellman, *New directions in cryptography*, *IEEE Transactions on Information Theory* **IT-22** (1976) 644-654.

[99] D. Goj, *Investigations in terminal security models for mobile communications networks*. Technical Report **CSD-TR-96-03**, Computer Science Department, Royal Holloway, University of London, February 1996.

[100] L. Chen, D. Gollmann, Y. Han and C. Mitchell, *Identity and location privacy in third generation mobile telecommunications*. To be submitted.

[101] M.J. Beller, L. Chang and Y. Yacobi, *Privacy and authentication on a portable communications system*, *IEEE Journal on Selected Areas in Communications* **11** (1993) 821-829.

[102] European Union Council Resolution, *International requirements for the lawful interception of telecommunications*. January 1995.

[103] N. Jefferies, *Security in third-generation mobile systems*. Presented at *IEE Electronics Division Colloquium on Security in Networks (London, February 1995)*. **IEE Digest No. 1995/024**.

[104] C.J. Mitchell, *Security in future mobile networks*. To appear in *Mobile multi-media* (C. I'Anson and T. Wilkinson, eds.), Prentice Hall.

[105] P. Syverson and P.C. van Oorschot, *On unifying some cryptographic protocol logics*. In *Proceedings of the 1994 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE Computer Society Press (1994) pp. 14-28.

[106] L. Chen and C. Mitchell, *Evaluation of a mutual authentication mechanism based on asymmetric techniques for UMTS.* Technical Report **CSD-TR-96-11**, Computer Science Department, Royal Holloway, University of London, 1996.

[107] L. Chen and D. Gollmann, *Formal verification of a mutual authentication protocol*. Technical Report **CSD-TR-96-05**, Computer Science Department, Royal Holloway, University of London, February 1996.

[108] M. Burrows, M. Abadi and R.M. Needham, *A logic for authentication*. DEC Systems Research Center Technical Report **39**, February 1989.

[109] L. Gong, R. Needham and R. Yahalom, *Reasoning about belief in cryptographic protocols*. In: *Proceedings of the 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pp.234-248. IEEE Computer Society Press, Los Alamitos, California, 1989.

[110] E. Snekkenes, *Exploring the BAN approach to protocol analysis*. In: *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp.171-181. IEEE Computer Society Press, Los Alamitos, California, 1991.

[111] R. Kailar and V.D. Gligor, *On belief evolution in authentication protocols.* In: *Proceedings of the Computer Security Foundations Workshop IV*, pp.103-116. IEEE Computer Society Press, Los Alamitos, California, 1991.

[112] C. Meadows, *A system for the specification and analysis of key management protocols*. In: *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp.182-195. IEEE Computer Society Press, Los Alamitos, California, 1991.

[113] C. Meadows, *Applying formal methods to the analysis of a key management protocol*. *Journal of Computer Security* **1** (1992) pp.5-35.

[114] P. Syverson, *Formal semantics for logics of cryptographic protocols*. In: *Proceedings of the Computer Security Foundations Workshop III*, pp.32-41. IEEE Computer Society Press, Los Alamitos, California, 1991.

[115] P. Syverson, *The use of logic in the analysis of cryptographic protocols.* In: *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, pp.156-170. IEEE Computer Society Press, Los Alamitos, California, 1991.

[116] P. Syverson and C. Meadows, *A logical language for specifying cryptographic protocol requirements*. In: *Proceedings of the 1993 IEEE Computer Society Symposium on Research in Security and Privacy*, pp.165-177. IEEE Computer Society Press, Los Alamitos, California, 1993.

13. Abbreviations

ANSI	Americal National Standards Institute (the U.S. member body of ISO)
AuC	Authentication Centre
BER	Bit Error Rate
BSS	Base Station Subsystem
CA	Certification Authority
CCF	Cryptographic Check Function (e.g. a MAC)
CDMA	Code-Division Multiple Access
CEIR	Central EIR
CLI	Calling Line Identifier
CODIT	COde-DIvision Testbed (a RACE project)
CRC	Cyclic Redundancy Check
DCA	Dynamic Channel Allocation
DCCH	Dedicated Control CHannel
DECT	Digital European Cordless Telephony
DES	Data Encryption Standard (a block cipher which is the subject of an ANSI standard)
DS	Direct Sequence
DSS	Digital Signature Standard (a NIST standard specifying the *Digital Signature Algorithm*)
DSSA	Distributed System Security Architecture
DTI	Department of Trade and Industry (a UK government department)
EIR	Equipment Identity Register
EPSRC	Engineering and Physical Sciences Research Council
ETSI	European Telecommunications Standards Institute
FDMA	Frequency-Division Multiple Access
FE	Functional Entity
FH	Frequency Hopping
FPLMTS	Future Public Land Mobile Telecommunications System
GMW	Gordon-Mills-Welch (a family of sequences)
GPS	Global Positioning System
GSM	Global System for Mobile communications
HLR	Home Location Register
IEC	International Electrotechnical Commission

ILP	Identity and Location Privacy
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IMUI	International Mobile User Identity
IN	Intelligent Network (a model for telecommunications networks)
ISO	International Organization for Standardization
ITU	International Telecommunications Union
KDC	Key Distribution Centre
KEK	Key Encrypting Key
KO	Key Offset
KTC	Key Translation Centre
LAI	Location Area Identifier
LFSR	Linear Feedback Shift Register
MAC	Message Authentication Code (see, for example, ISO/IEC 9797)
MBS	Mobile Broadband System (a RACE project)
MCF	Mobile Control Function
ME	Mobile Equipment
MSC	Mobile Switching Centre
MSF	Mobile Storage Function
NCC	National Computer Centre (a UK body based in Manchester)
NIST	National Institute for Standards and Technology (a U.S. Federal standards body)
NO	Network Operator
NOID	Network Operator IDentity
OSI	Open Systems Interconnection
PN	Pseudo-Noise
PRMA	Packet-Reservation Multiple Access
RACE	Research and development into Advanced Communications in Europe
RHUL	Royal Holloway, University of London
RI	Real Identity
RSA	Rivest-Shamir-Adleman (a public key cryptosystem and digital signature scheme)
SCF	Service Control Function
SDF	Service Data Function
SDMA	Space-Division Multiple Access

SFH	Slow Frequency Hopping
SP	Service Provider
SSMA	Spread Spectrum Multiple Access
SVO	Syverson-Van Oorschot (a logic designed to test authentication protocols)
TCH	Traffic CHannel
TDMA	Time-Division Multiple Access
TI	Temporary Identity
TMSI	Temporary Mobile Subscriber Identity
TMUI	Temporary Mobile User Identity
TR1	Technical Report 1 (of this project)
TR2	Technical Report 2 (of this project) = this report
TR3	Technical Report 3 (of this project)
TTP	Trusted Third Party (e.g. an authentication server, a certification authority, etc.)
TVP	Time Variant Parameter (i.e. a timestamp, nonce or sequence number)
UIM	User Identity Module
UMTS	Universal Mobile Telecommunications System
VLR	Visitor Location Register
XOR	eXclusive OR
3GS	3rd Generation mobile telecommunications System
3GS3	3rd Generation mobile telecommunications System Security Studies (the title of this project)

Appendix A: Document history

Release dates

The first version of this document was released to the LINK PCP Committee in mid October 1994, the second version was released in mid September 1995, and this, the final version, was released on May 15th 1996 (shortly after completion of the project).

Document history

Date	Version	Changes
15th July 1994	Draft A	Initial draft.
4th September 1994	Draft B	Major revision by RHUL based on comments from all partners.
22nd September 1994	Draft C	Revision by RHUL based on comments from all partners.
15th October 1994	Version 1	Minor revision by RHUL based on comments from Vodafone.
12th July 1995	Draft A of Version 2	Initial draft of Version 2. Prepared by RHUL.
23rd July 1995	Draft B of Version 2	Revision by RHUL incorporating two RHUL internal reports.
23rd August 1995	Draft C of Version 2	Major revision by RHUL incorporating various internal documents, together with comments from Vodafone.
20th September 1995	Version 2	Minor revision by RHUL based on comments from RHUL and Vodafone.
19th December 1995	Draft A of Version 3	Initial draft of Version 3. Prepared by RHUL.
4th January 1996	Draft B of Version 3	Revision by RHUL.
6th February 1996	Draft C of Version 3	Revision by RHUL incorporating various internal documents, together with comments from Vodafone.
12th May 1996	Draft D of Version 3	Revision by RHUL incorporating various internal documents, together with comments from Vodafone.
15th May 1996	Version 3 (Final version)	Minor revision by RHUL.