CRYPTOGRAPHIC TECHNIQUES FOR PERSONAL COMMUNICATION SYSTEMS SECURITY

by

John Christopher Cooke Esq., B. Eng (Hons), AMIEE, MIEEE.

Submitted for the degree of
DOCTOR OF PHILOSOPHY

at

The University of Aston in Birmingham,
May, 1995.

CRYPTOGRAPHIC TECHNIQUES FOR PERSONAL COMMUNICATION SYSTEMS SECURITY

by

John Christopher Cooke, B. Eng (Hons), AMIEE, MIEEE

Submitted for the degree of
DOCTOR OF PHILOSOPHY
at
The University of Aston in Birmingham
1995

SUMMARY

The advent of personal communication systems within the last decade has depended upon the utilization of advanced digital schemes for source and channel coding and for modulation. The inherent digital nature of the communications processing has allowed the convenient incorporation of cryptographic techniques to implement security in these communications systems. There are various security requirements, of both the service provider and the mobile subscriber, which may be provided for in a personal communications system. Such security provisions include the privacy of user data, the authentication of communicating parties, the provision for data integrity, and the provision for both location confidentiality and party anonymity.

This thesis is concerned with an investigation of the private-key and public-key cryptographic techniques pertinent to the security requirements of personal communication systems and an analysis of the security provisions of Second-Generation personal communication systems is presented. Particular attention has been paid to the properties of the cryptographic protocols which have been employed in current Second-Generation systems.

It has been found that certain security-related protocols implemented in the Second-Generation systems have specific weaknesses. A theoretical evaluation of these protocols has been performed using formal analysis techniques and certain assumptions made during the development of the systems are shown to contribute to the security weaknesses. Various attack scenarios which exploit these protocol weaknesses are presented.

The Fiat-Shamir zero-knowledge cryptosystem is presented as an example of how asymmetric algorithm cryptography may be employed as part of an improved security solution. Various modifications to this cryptosystem have been evaluated and their critical parameters are shown to be capable of being optimized to suit a particular application. The implementation of such a system using current smart card technology has been evaluated.

Various recommendations are made regarding the development of security functionality and the implementation of security in Third-Generation personal communication systems.

KEY WORDS

AUTHENTICATION, FORMAL LOGIC, PROTOCOLS,
SYSTEM OVERHEADS, ZERO KNOWLEDGE.

This thesis is dedicated to my mother, Phyllis,

and to the dear memory of my father, Roland,

for all their love, support and encouragement,

and also

to the dear memory of my grandparents,

Doris and Donald Kitcher Esq., MBE.

In research the horizon recedes as we advance,
and is no nearer at sixty than it was at twenty.
As the power of endurance weakens with age,
the urgency of the pursuit grows more intense
...And research is always incomplete.

Mark Pattison (1813 - 1884)
Asaac Casaubon (1875, ch. 10)

Sed quis custodiet ipsos Custodes?

Juvenal (C. 100 AD)

Post equitem sedet atra Cura.

Horace (65 - 8 BC)
Odes (III. i .40)

ACKNOWLEDGEMENT

I wish to thank my supervisor, Dr. Ron Brewster, for his continued support and guidance during my period of research and thesis writing,

and also

Dr. Richard French, BNR (Europe) Ltd., for his kind support of this research project.

LIST OF CONTENTS

LIST OF FIGURES AND TABLES

Page Number

List of Figures

List of Tables

LIST OF ABBREVIATIONS

AC	Authentication Code
ADC	ADministration Centre
ADPCM	Adaptive Differential Pulse Code Modulation
ANSI	American National Standards Institute
ASN	Abstract Syntax Notation
AUC	AUthentication Centre
BAN	Burrows, Abadi and Needham (formal logic)
BS	Base Station
BSC	Base Station Controller
BTS	Base Transceiver Station
CAI	Common Air Interface standard (ETSI)
CBC	Cipher Block Chaining
CCCH	Common Control CHannel
CCIR	Comité Consultatif International de Radio (ITU)
CCITT	Comité Consultatif International des Télégraphes et Téléphones (ITU)
CDMA	Code Division Multiple Access
CFM	Cipher Feedback Mode
CFSM	Cryptographic Finite State Machine
CK	Ciphering Key
CUG	Closed User Group
DAM	DECT Authentication Module
DC	Domestic Cordless
DCK	Derived Ciphering Key
DCS	Digital Cellular System
DEA	Data Encryption Algorithm
DECT	Digital European Cordless Telephone
DES	Data Encryption Standard
DFT	Discrete Fourier Transform
DIVU	DIVision Unit
DSP	Digital Signal Processing
ECB	Electronic Code Book
EEPROM	Electrically Erasable Programmable Read Only Memory
EIR	Equipment Identity Register
EPROM	Erasable Programmable Read Only Memory
ETSI	European Telecommunications Standards Institute
FACCH	Fast Associated Control CHannel
FEAL	Fast data Encipherment ALgorithm

FPLMTS	Future Public Land Mobile Telecommunications System (CCIR)
FT	Fixed Terminal (DECT)
FP	Fixed Part (DECT BS)
GCD	Greatest Common Denominator
GMSK	Gaussian filtered Minimum Shift Keying
GSM	Groupe Spéciale Mobile (ETSI)
HLR	Home Location Register
I/C	Incoming
IAM	Incoming Attention Message
IC	Integrated Circuit
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Initial Permutation
ISDN	Integrated Services Digital Network
ISO	International Standards Organization
ITU	International Telecommunication Union
IV	Initial Vector (or Value)
K	authentication Key (DECT)
Kc	ciphering Key (GSM)
Ki	identification Key (GSM)
KMC	Key Management Centre
KS	Key for Session authentication (DECT)
KS'	Key for reverse Session authentication (DECT)
LAI	Location Area Identification
LAPDm	Link Access Procedure, D-channel, (frame) multiplexing - (CCITT - ISDN, Data Link Layer protocol)
LCM	Lowest Common Multiple
LHS	Left Hand Side
LTP	Long-Term Prediction
MAC	Message Authentication Code / Medium Access Control
MOD	MODulo arithmetic
MS	Mobile Station
MSC	Mobile services Switching Centre
MULU	MULtiplication Unit
MUX	Multiplexer
NMC	Network Management Centre
NPN	Numerical Petri Nets
NSA	National Security Agency (US)
O/G	Outgoing
OFM	Output Feedback Mode

OMC	Operation and Maintenance Centre
OSI	Open Systems Interconnection (ISO, 7-layer model)
PABX	Private Automatic Branch eXchange
PCN	Personal Communication Network
PCS	Personal Communication System
PIN	Personal Identification Number
PKP	Permuted Kernel Problem
PLMN	Public Land Mobile Network
PP	Portable Part (of DECT PT)
PRGF	Pair-wise Relatively-prime Generation Function
PSTN	Public Switched Telephone Network
PT	Portable Terminal (DECT)
PTD	Power-of-Two-Distance
PTO	Public Telecommunications Operator
RACE	Research and development in Advanced Communication technologies in Europe
RAM	Random Access Memory
RAND	Random vector (GSM)
RAND_F	Random value, generated by FT, for authentication of PT (DECT)
RAND_P	Random value, generated by PT, for authentication of FT (DECT)
RES1	RESult of authentication of PT (DECT)
RES2	RESult of authentication of FT (DECT)
RF	Radio Frequency
RHS	Right Hand Side
ROM	Read Only Memory
RPE	Regular Pulse Excitation
RS	authentication value (DECT)
RSA	Rivest, Shamir and Adleman public key algorithm
SACCH	Slow Associated Control CHannel
SCK	Static Cipher Key (DECT)
SDCCH	Stand-alone Dedicated Control CHannel
SDL	functional Specification and Description Language
SIM	Subscriber Identity Module
SL	Security Logic
SMG5	Special Mobile Group 5 (ETSI - UMTS)
SRES	Signed RESponse (GSM)
SS	Supplementary Services (management)
SS7	Signalling System no.7 (CCITT)
TDMA	Time Division Multiple Access
TMSI	Temporary Mobile Subscriber Identity (GSM)
TOF	Trapdoor One-way Function

UAK	User Authentication Key (DECT)
UIM	User Interface Module (UMTS)
UMTS	Universal Mobile Telecommunication System (RACE)
UPI	User Personal Identity (DECT)
US	United States (of America)
VLR	Visitor Location Register
XOR	eXclusive-OR
XRES1	eXpected RESult of authentication of PT (DECT)
XRES2	eXpected RESult of authentication of FT (DECT)

CHAPTER 1. INTRODUCTION TO COMMUNICATIONS SECURITY.

This chapter presents an overview of the rationale behind the provision of security-related functionality in communications systems. The various threats of attack upon a communications system are examined and the various security requirements of both the users of (public) telecommunications systems and the service providers are categorized.

The fundamental techniques of incorporating security (privacy) in both analogue and digital communication systems are discussed and an overview of the generic types of cipher system is presented. Similarly, party authentication schemes and data integrity schemes are also considered.

The various generic properties pertinent to the incorporation of cryptosystems within communication systems are also considered, including security level, cryptographic key management, cryptosynchronization, error propagation, data bandwidth preservation, and the system overheads associated with processing and signalling.

1.1 The threat of attack upon communication systems

All communications systems, whatever their nature, are susceptible to attack. An attack upon a communication system can be defined as an attempt by an authorized or unauthorized party to gain access to data communicated by the system.

An attacker of a system may wish to attempt an attack for several reasons :-

• Recovery of user data. An attacker may wish to recover user data in order to be privy to whatever data is communicated between the parties. Such an attack is generally referred to as "eavesdropping".

• Modification of user data. An attacker may wish to modify data generated by users of a communication system perhaps in order to attempt an impersonation (possibly to gain a pecuniary advantage) or perhaps to cause confusion or to prevent communications. An attacker may also wish to erase user data in order to cause disruption of communications, or to add to the user data.

• Recovery of signalling information. An attacker may wish to recover signalling information associated with a communications link in order to determine the identity of the communicating parties or their locations.

• Modification of signalling information. An attacker may wish to modify signalling information associated with a communications link in order to avoid billing or to cause disruption of communications, or to attempt an impersonation.

• Recovery of traffic intelligence. An attacker may wish to recover traffic intelligence; the flow of traffic may convey information relating to the activity of the communicating parties.

1.1.1 The nature of attack upon communication systems.

Any attack upon a communications system can be categorized as being either a passive attack or an active attack, either a direct attack or an indirect attack, and either a fortuitous attack or an occasioned attack.

1.1.1.1 Passive attack. A passive attack is one in which an attacker does not make changes of any kind to the flow of data in a system; no data is modified or removed, and no data is supplied. In other words, a passive attack is one in which data is only monitored.

1.1.1.2 Active attack. An active attack is one in which an attacker makes changes to the data being communicated within a system (be it either data modification or removal), or inputs additional data to the system.

1.1.1.3 Direct (local) attack. A direct attack is one where some weakness in a communication system is exploited locally . An example of a direct attack is a wire-tap on a communications line.

1.1.1.4 Indirect (remote) attack. An indirect attack is where some weakness in a communication system is exploited remotely. An example of an indirect attack is the exploitation of compromising RF emissions from an item of communications equipment.

1.1.1.5 Fortuitous attack. A fortuitous attack is where an intrinsic weakness in a communications system is exploited. An example of such an intrinsic weakness is the communication of user data in clear (unprotected) form across the air interface.

1.1.1.6 Manipulative attack. A manipulative attack is where the system is manipulated to allow the exploitation of a weakness. An example of a manipulative attack is the "chosen ciphertext attack" on a cryptosystem; refer to Section 1.2.3.3.

1.1.2 Vulnerability of the target.

The threat of attack upon a particular communication system is directly related to the vulnerability to attack of that communication system. The vulnerability to attack of a communication system is dependent upon the physical security and the logical security with which the data that is stored and / or communicated in the system is protected. The physical security protecting a communications system may include such aspects as the protection against compromising emissions (*eg*. RF shielding), and a tamper-resistant module such as an IC card (or smart card). The logical security protecting a communications system may include such aspects as party authentication (*eg*. password-controlled access), and the intrinsic protection of user data, typically by scrambling (in analogue systems) or by encryption (in digital systems).

1.1.3 The nature of transmitted data.

The nature of data transmitted in a communication system also has a direct bearing upon the threat of attack upon the system. Apart from the intrinsic nature of the transmitted data, be it user data or signalling information (see above), the data may represent several different types of user, whom may require varying degrees of protection by virtue of their nature of business. For example, a "general public" user of a telecommunications system may well require only a relatively low level of protection of user data sufficient, say, to deter casual eavesdropping, whereas a commercial user may well require a relatively high level of protection of user data in order to provide a guaranteed level of protection from deliberate attacks, which may be both protracted and sophisticated.

Typical types of commercial user data that require a high level of protection are medical, legal, financial, sales, research and development. It is the varying requirement for the level of protection to be provided by the security measures that are implemented, that has led to the concept of *cover time*, refer to Beker and Piper [1]. This is the time during which the data protected by the security measures is considered to be secure from any successful attack.

1.2 Means of attacking cryptographically protected communication systems.

There are several ways in which a cryptographically protected communications system may be subject to attack, such attacks fall into three broad categories.

1.2.1 Brute force attack.

A brute force attack is one in which an attacker searches the entire cryptographic key space until the correct key is found. It is called a brute force attack because it does not require any particular knowledge about the cryptovariable being used, but requires the ability to try all the possible permutations for a particular length of cryptovariable, given knowledge (or assumptions) about the particular cryptographic algorithm employed. This will result in a trade-off between the required processing power and the time required to perform the calculations. Any realistic cryptosystem will inevitably require substantial amounts of both processing power and time in order for a brute force attack to be mounted. Some cryptosystems may prevent certain brute force attacks by means of protocols which prevent repetitive attempts to use a key in a given period of time *eg* Bank automatic teller machines prevent any more than three (typically) attempts to enter the correct PIN.

1.2.2 Masquerade attack.

A masquerade or "spoof" attack is one in which an attacker poses as a legitimate entity, in order to either gain access to the particular data belonging to that user or to gain access to the communication system in general, refer to Seberry and Pieprzyk [2]. There are many different forms of masquerade attack, but they essentially fall within two broad categories.

The first type of masquerade attack is where an attacker possesses little or no knowledge about a cryptosystem. Communications between legitimate entities of the cryptosystem may be intercepted by an attacker, who may attempt to replace current messages with those previously used in an attempt to cause the system to repeat whatever resulted from the previous instance of the transmission of such messages. Such a "blind" masquerade attack may be attempted during an authentication exchange, where weak cryptosystems may transmit the same data to achieve a particular authentication. This form of masquerade attack is also known as a protocol reflection attack, refer to Figure 1.1.

The second type of masquerade attack is where an attacker possesses specific knowledge about a cryptosystem such that it is possible to impersonate an entity of the system in order to mount an active attack where data is input to the system which has not necessarily been generated by the interception of previous communications. In such an attack data may be input to the system which has been calculated from the specific knowledge of the system and is used in an attempt to produce a desired result. The desired result could be, say, a false authentication or it could be the recovery of specific data from the system which would enable particular calculations to be made in order to supply further data to the system. The latter would constitute an iterative masquerade attack, refer to Figure 1.2.

1.2.3 Cryptographic (statistical) attack.

A cryptographic attack is where an attacker attempts to gain knowledge about the cryptosystem and / or the transmitted data (be it user data or signalling information) by statistical analyses of the transmitted data. This may or may not be complemented by a prior knowledge of details of the particular cryptographic algorithm(s) employed. Such statistical analyses may be categorized as follows.

1.2.3.1 Ciphertext-only attack. In such an attack the attacker has knowledge only of the ciphertext that has been intercepted. There is no knowledge of the plaintext corresponding to the ciphertext, although there may be some knowledge about the algorithms employed in the cryptosystem concerned. Any such attack on a cryptosystem would have to rely purely on a statistical analysis of whatever ciphertext was recovered, perhaps on the basis of assumptions regarding the nature of the corresponding plaintext.

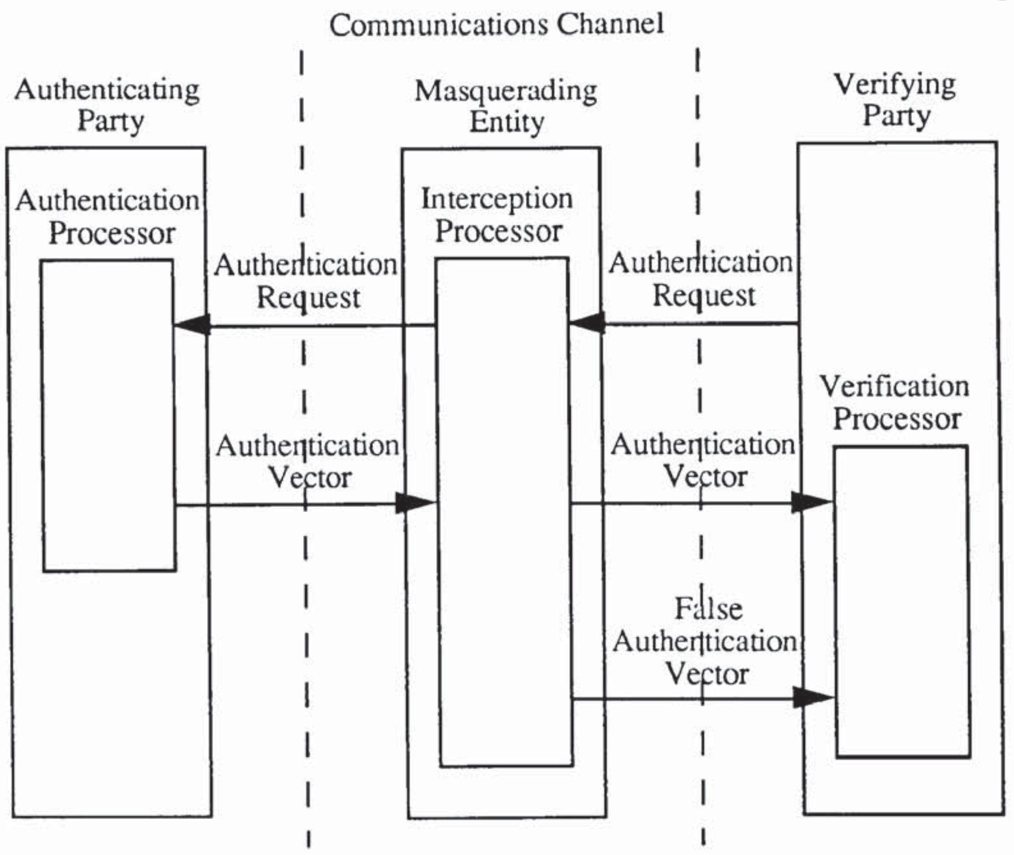


Figure 1.1 Protocol Reflection Attack

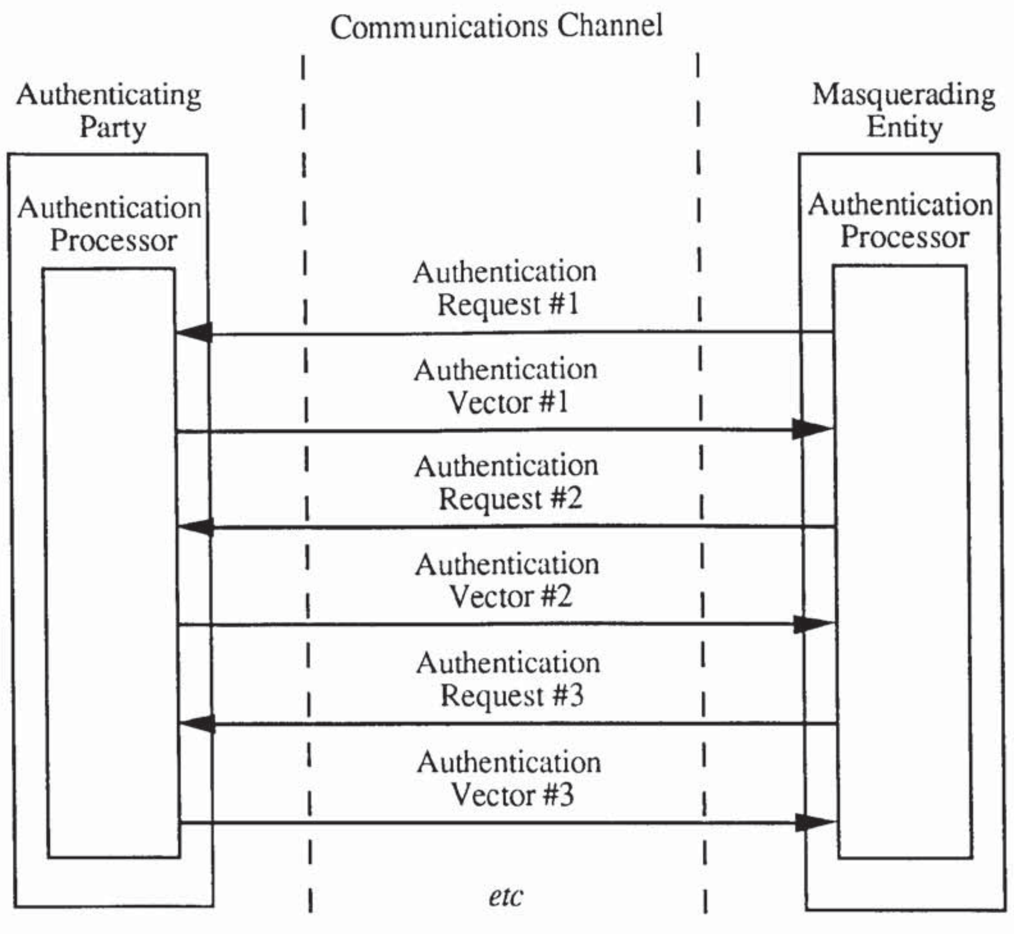


Figure 1.2 Iterative Masquerade Attack

1.2.3.2 Known-plaintext attack. In such an attack the attacker has knowledge both of the recovered ciphertext and some of or all of its corresponding plaintext. Such knowledge would afford the attacker greater opportunity for statistical analysis of the cryptographic algorithm(s) employed by the cryptosystem, of the cryptographic key(s) used, and of any ciphertext to which the corresponding plaintext is unknown.

1.2.3.3 Chosen-plaintext attack. In such an attack the attacker not only has knowledge both of the ciphertext and of its corresponding plaintext, but also is able to choose what the plaintext shall be. This would afford the attacker the greatest opportunity for statistical analysis of the cryptographic algorithm(s) employed by the cryptosystem and of the cryptographic key(s) used, and would hence lead to a possible situation of the attacker being able to recover plaintext from a subsequent ciphertext-only attack on the same cryptosystem, perhaps not even using the same cryptographic key(s).

1.3 Security in Analogue Communication Systems.

Analogue communication systems suffer from two intrinsic disadvantages with regard to their communications security; they are easy to eavesdrop upon, and they are difficult to protect with a high level of security without suffering from associated problems.

1.3.1 Eavesdropping in analogue systems.

The reason why analogue communications are so easy to eavesdrop upon is that they have not (in their unmodified form) been subjected to any form of source coding. This means that the communicated analogue data has a high degree of redundancy and requires no decoding for recovery. Additionally, there is no synchronization requirement between a transmitter and any receiver.

In order to attempt to protect analogue communications with a high level of security, it is necessary to perform certain transformations on the data to render it unintelligible to any party not authorized to receive it. The various techniques which may be employed are generically referred to as analogue scrambling. It should be noted that although the scrambling techniques implemented are analogue in effect, they may be achieved both by analogue and by digital signal processing. It is the latter form which is favoured in modern analogue scrambling systems, due to the higher levels of security, precision, automation and flexibility achievable, refer to Pichler [3].

1.3.2 Analogue scrambling techniques.

Analogue speech scramblers transform analogue speech (often by manipulation of a digital form of the signal). A speech signal may be regarded as a three-dimensional quantity with amplitude, frequency and time defining the dimensions. The scrambling process may, therefore, be regarded as a transformation of a surface in a 3-dimensional space.

Voice quality is degraded by these operations to some extent, this being particularly true in the case where there are combined operations in different domains. These are known as multi-dimensional scramblers, refer to Brunner [4].

Analogue scramblers are, for convenience, sub-divided into three basic categories: frequency domain scramblers, time domain scramblers and multi-dimensional scramblers. The relative merits of these different systems are broadly considered below. Note that these scramblers may all be considered to incorporate some scrambling in the amplitude dimension, hence amplitude scramblers are not separately considered. Refer to Beker and Piper [5] for full details of analogue scrambling schemes. Figure 1.3 shows a typical configuration of an analogue scrambling transceiver.

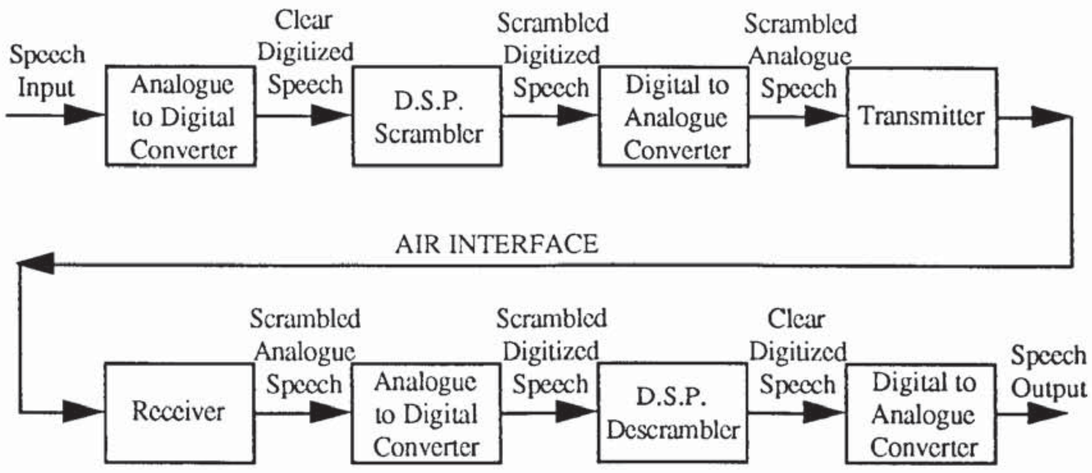
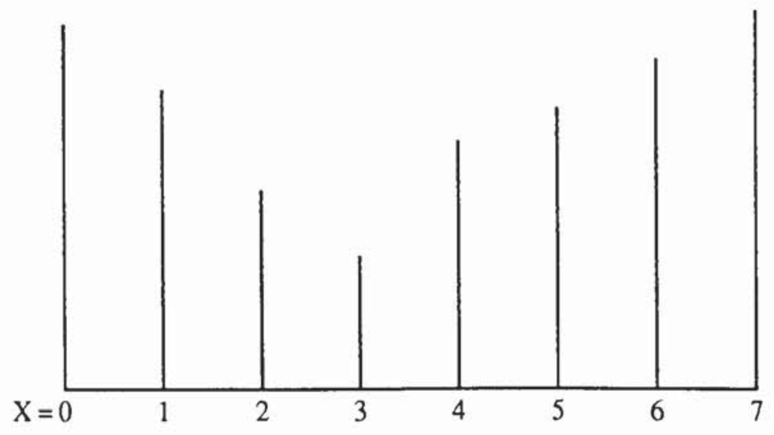

Speech Input
Analogue to Digital Converter
Clear Digitized Speech
D.S.P. Scrambler
Scrambled Digitized Speech
Digital to Analogue Converter
Scrambled Analogue Speech
Transmitter
AIR INTERFACE
Receiver
Scrambled Analogue Speech
Analogue to Digital Converter
Scrambled Digitized Speech
D.S.P. Descrambler
Clear Digitized Speech
Digital to Analogue Converter
Speech Output

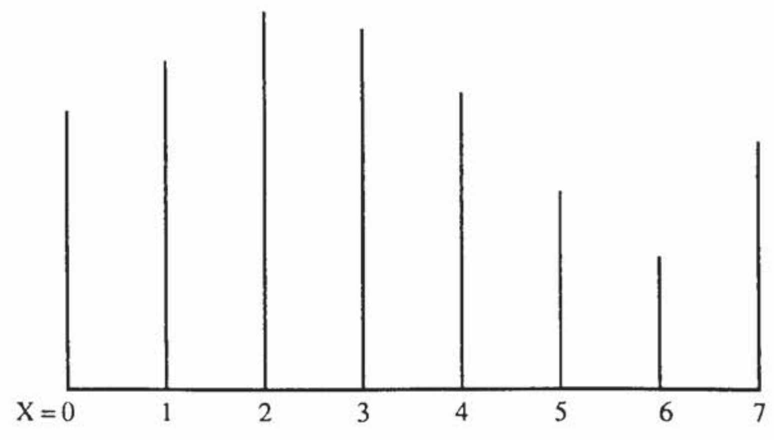
Figure 1.3 An Analogue Scrambling Transceiver Using D.S.P.

1.3.2.1 Frequency domain techniques. This describes a class of scramblers which operate purely in the frequency domain. Their scrambling operation modifies the frequency components of the speech. They have a number of advantages; in particular they can exhibit a low level of residual intelligibility, they generally give good recovered audio quality, only a short system time delay is introduced and the system is normally available at a low cost. The main disadvantages of such systems are that they often offer the lowest level of security (particularly so for simple frequency inverters), and the permuted frequency spectrum may well necessitate modifications to the transmission channel (particularly with respect to its bandwidth and signal-to-noise ratio). Most analogue frequency scramblers are useful only as privacy devices to deter casual eavesdropping. Figure 1.4 illustrates how a frequency domain scrambler works, showing frequency inversion using a DSP-based discrete Fourier transform.

1.3.2.2 Time domain techniques. This describes a class of scramblers which operate purely in the time domain. Their scrambling operation divides the speech into time slots, (called segments), and transmits them in a temporally permuted order. These can also yield a low level of residual intelligibility, but they are resistant to many of the attacks which succeed against frequency scramblers, refer to Hong and Keubler [6]. Time element scramblers preserve the speech signal spectrum, although they introduce a noticeable time delay into the transmission path that can cause problems for full duplex speech communications which, if the total time delay from speaker to listener exceeds approximately 100 ms, may result in "double speak" , refer to Section 1.3.4.4. Their use can also lead to a significant degradation in audio quality due to discontinuities being introduced in the audio signal, (refer to Section 1.3.4.3). Figure 1.5 illustrates how a time domain scrambler works, showing reversed time segmentation.



Typical D.S.P. Line Spectrum (n=8)



Corresponding Frequency Inverted D.F.T. Spectrum

Figure 1.4 Frequency Inversion Using D.S.P.

1.3.2.3 Two-dimensional techniques. This describes a class of scramblers which operate in more than one dimension, normally in both the time and the frequency domains. They normally employ either a time element scrambler in conjunction with some fairly simple frequency scrambling or, alternatively, a frequency scrambler in conjunction with some fairly simple time element scrambling. Since time element scrambling is, generally speaking, potentially the stronger of these two options, the former kind is generally to be preferred, refer to Jayant *et al* [7]. A well designed multi-dimensional scrambler yields a very low level of residual intelligibility and is generally resistant to all but the most sophisticated attacks on analogue systems. Unfortunately, not all scramblers which claim to be multi-dimensional and highly secure justify such a claim. Poorly designed devices which, for example, employ a linear sequence generator for selection of the scrambling algorithm, can lull the user into a false sense of security. The drawbacks of multi-dimensional scramblers are similar to those of time element scramblers.

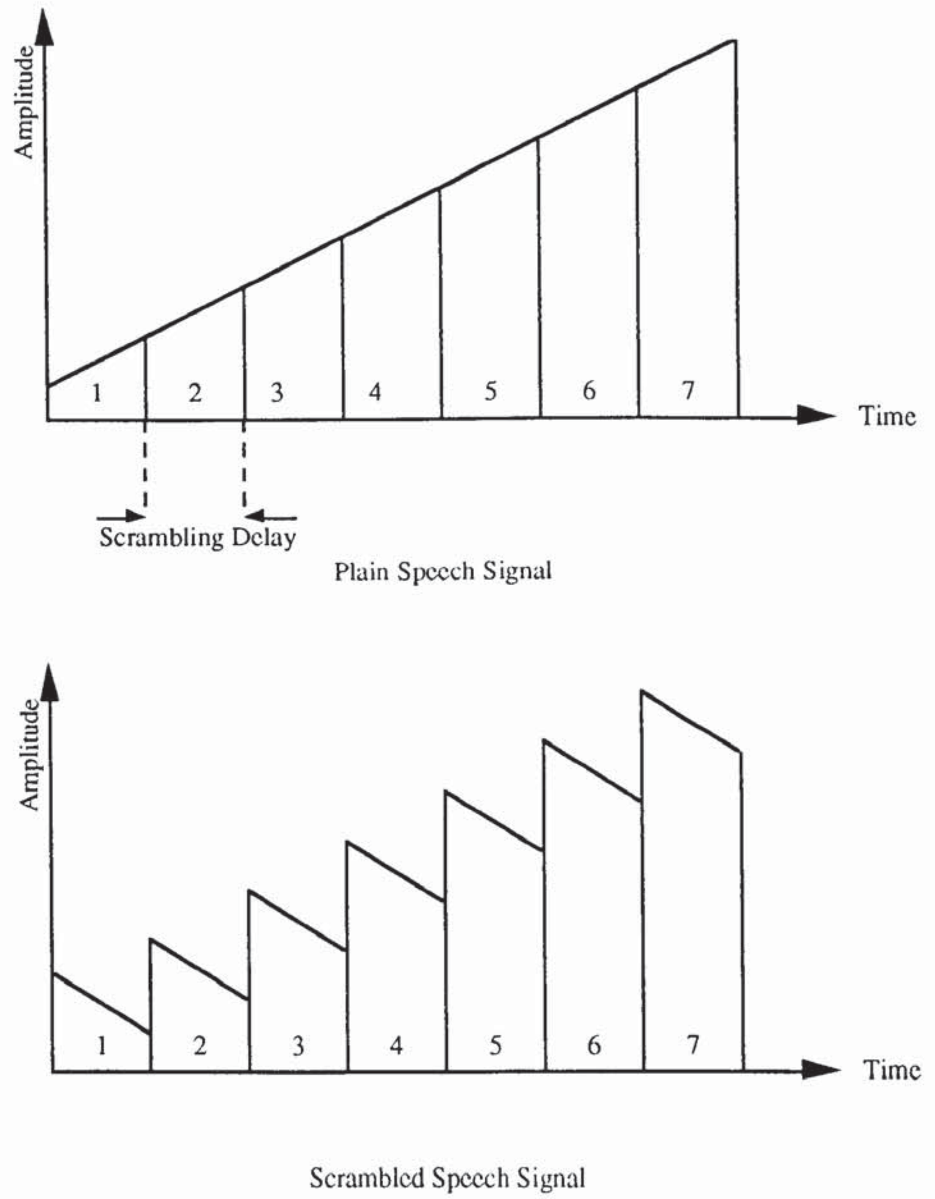


Figure 1.5 Time Segment Permutation

1.3.3 Noise masking.

This aspect of analogue communications has been included because it is an important aspect in analogue communications security which is invariably omitted in works regarding the subject.

All analogue communications suffer from a certain degradation of received signal quality due to noise contamination. The noise may be generated internally by the communications equipment or it may be received from outside the equipment, and the noise can be of several types. As the signal-to-noise ratio decreases, it becomes progressively harder to recover the wanted analogue signal.

Noise masking is a technique of deliberately introducing noise into the analogue communications channel, in order to create difficulty in recovering the signal. The problem of merely introducing such noise is that the associated difficulty in signal recovery would be experienced by all parties. A particular use of noise masking is where two physical conductors, which constitute one communications line, need to be protected because of a risk of unwanted electromagnetic coupling into neighbouring conductors or because of a risk of such electromagnetic radiation being recovered by a local antenna.

There are many cases where one physical communications channel consists of two conductors made physically close (such as twisted pair or twin-axial cable) and the signals applied to the conductors are in anti-phase (from a balanced, differential source). Common examples are in telephone communications and local area networks.

Such cables tend to reduce electromagnetic radiation because of the anti-phase effects of the neighbouring conductors, although there is always some residual radiation even if electromagnetic screening is employed. Noise masking can be useful here, since an identical noise signal may be added to both conductors in phase at the send end of such a cable, but be rejected at the receive end by means of either a balanced transformer or an electronic differential amplifier with a high common-mode rejection such as an operational amplifier. Note that the noise is cumulative with regard to electromagnetic radiation and so it can be used to mask any stray radiation of the communicated signal. Figure 1.6 illustrates a typical arrangement for the inclusion of noise masking on a data communications cable.

1.3.4 Weaknesses in analogue scrambling systems.

The choice of an analogue scrambler must (as is true for all scramblers) involve a trade-off between security, price and performance. The security level may be considered to be paramount, but there may be limitations imposed with regard to its performance in such areas as audio signal distortion and processing time delay. These are considered below.

1.3.4.1 Vulnerability to intuitive attack. Analogue scrambling systems were developed to protect analogue communications in the form of speech or facsimile data, both of which are susceptible to intuitive attack. An intuitive attack is one in which the audio and/or image processing capability of the human brain may be used to "unscramble" patterns and to adapt to modified rules of processing. An intuitive attack upon such a system can be made visually in the case of scrambled speech or facsimile data, and can also be made acoustically in the case of scrambled speech.

In order to effect a visual intuitive attack it is first necessary to transform the scrambled data into a visual form. In the case of a facsimile this is simply a plot, but speech requires that a spectrograph is produced, this being a three-dimensional plot of frequency and amplitude against time. An attempt may then be made to reconstruct the original unscrambled data by treating the plot as a "jigsaw puzzle", refer to Beker and Piper [8].

An acoustic intuitive attack may be attempted upon analogue speech data by careful listening to the scrambled speech. It is possible to train the human brain to translate analogue speech which has been scrambled (particularly in the frequency domain) into plain speech.

It is true that such intuitive attacks are somewhat hit-and-miss in their effect, but they are successful to a varying degree; many analogue scrambling schemes are far from rigourous in their operation. As analogue scrambling schemes are made more complex they require automated operation *i.e.* they require the use of a microprocessor. It should be noted that analogue scramblers become more effective in securing signals as the plain signal is divided into smaller segments; the smallest segments possible being equivalent to digitization of the analogue signal. Thus it can be said that analogue scramblers tend towards digital scramblers when higher levels of security are attempted.

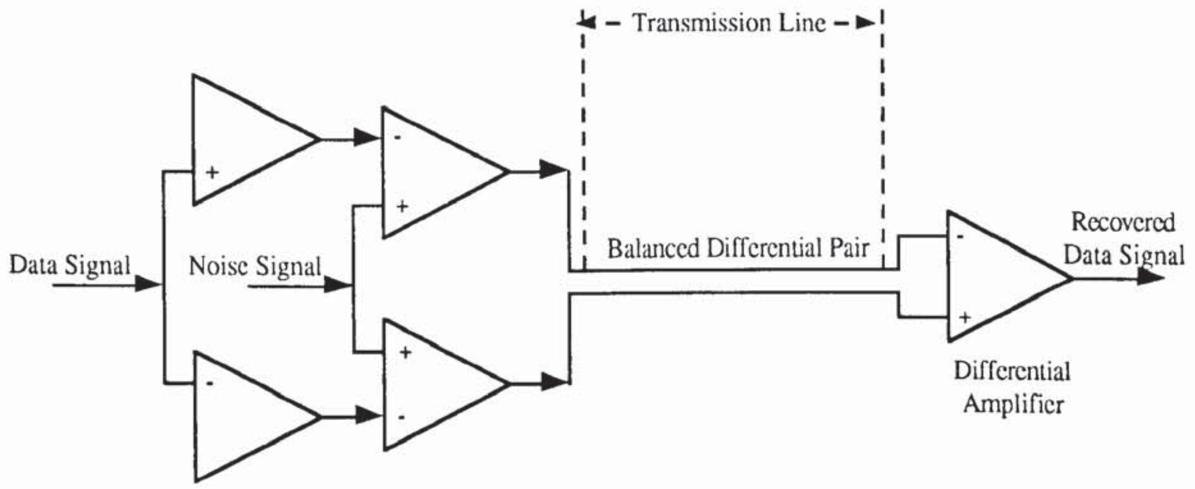


Figure 1.6 Addition of Noise Masking on a Transmission Line

1.3.4.2 Exploitation of analogue hardware weaknesses. Analogue hardware is susceptible to exploitation of any intrinsic weaknesses in its security. The most exploitable weakness is typically the susceptibility to intuitive attack, as explained above. There may also exist, however, other potentially exploitable hardware weaknesses such as unwanted coupling of the plain signal into data, control or power lines, and electromagnetic radiation related to the plain signal.

1.3.4.3 Audio signal distortion. A particular problem associated with analogue signal scramblers is the resultant distortion of the analogue signal upon recovery. This is caused by problems associated with the bandwidth in which the signal is transmitted. When an analogue signal is subjected to forms of frequency translation and/or inversion, parts of the signal which convey more information than others may be transmitted in parts of the channel pass-band which have a poorer frequency response, a greater delay characteristic or a lower signal-to-noise ratio. When an analogue signal is subjected to any form of time segment permutation, discontinuities are created in the scrambled signal which may not be successfully transmitted within the limited pass-band of the available channel.

1.3.4.4 Signal processing time delay. The time taken to process signals can be an important factor, particularly in duplex speech communications, where delays in excess of approximately 100 ms can be highly distracting to the communicating parties and can lead to symptoms such as "double speak" where both communicating parties inadvertently talk together. This is particularly significant in analogue speech scramblers employing time segment permutation, since it limits the length of any permutation frame to a maximum of 100 ms and hence puts a maximum bound on the level of security achievable by using this technique, due to the maximum time segment delay achievable .

1.4 Security in Digital Communication Systems

The implementation of secure communications in digital systems is considered below in terms of the susceptibility of such systems to casual eavesdropping, the properties of algorithms for encryption and for authentication, the recovery of traffic intelligence, the use of hybrid cryptosystems, and the exploitation of any intrinsic weakness in digital cryptosystem hardware.

1.4.1 Eavesdropping in digital systems.

Eavesdropping in digital communication systems is generally a more difficult task than it is in analogue systems because the transmitted data will not be directly intelligible to the human brain and thus will not be susceptible to any form of intuitive attack. More importantly, if the digital data represents some form of analogue data (such as speech), it will invariably be the result of some form of source coding, resulting in very little intrinsic redundancy. There will thus be a need to decode the digital data recovered, requiring both a knowledge of the encoding algorithm used and an ability to achieve bit synchronization. Furthermore, there may be channel coding employed in order to provide error detection and correction, hence introducing artificial redundancy.

Thus it can be seen that digital communications are inherently more difficult to eavesdrop upon than are analogue communications. This security advantage is surpassed by the fact that digital data is ideally suited for encryption. This is because such data consists of discrete, uniform segments which are of the smallest size and are binary in nature.

It should be noted that the generic term "encryption" refers to the process of transforming plain data (or "plaintext") into encrypted data (or "ciphertext") by means of a cryptographic algorithm (or "cipher"). Such a cryptographic algorithm requires a cryptographic key (or "cryptovariable") for the purposes of encryption or decryption, as opposed to a code which requires no key.

There follows a brief explanation of cryptographic techniques. A detailed appraisal of cryptographic techniques pertinent to the area of study, including their uses, implementation and weaknesses, is given in Chapter Two. Figure 1.7 shows a typical configuration of a digital encryption transceiver.

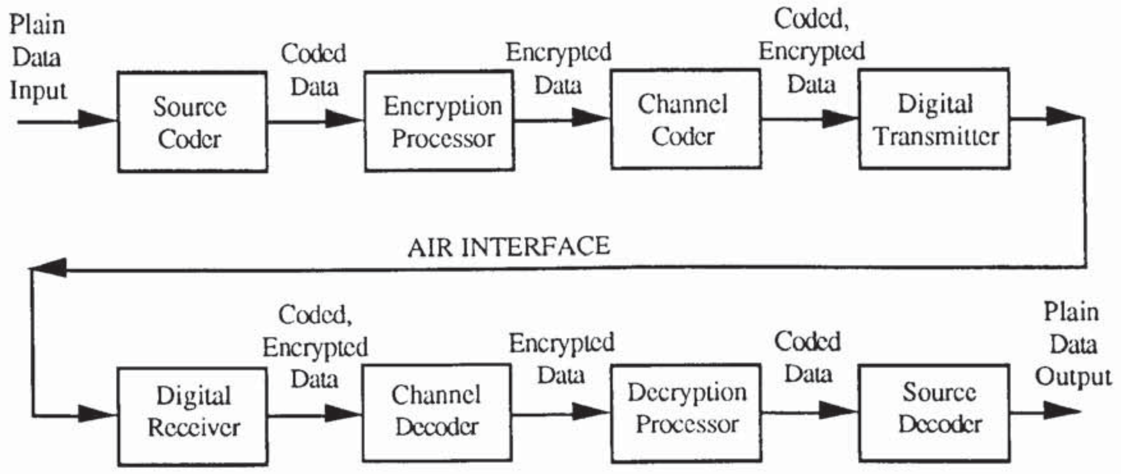


Figure 1.7 A Digital Encryption Transceiver

1.4.2 Digital encryption techniques.

There are many different types of encryption technique, which may be categorized in several ways. Each type has its associated advantages and disadvantages. There follows a general introduction.

1.4.2.1 Alphabetic ciphers. Alphabetic ciphers are the longest established types of cipher. They employ techniques of changing the representation of an alphabetic message such that members of a particular alphabet are represented by members of another alphabet (or other alphabets), where the latter may be of a fixed or a varying nature. There are two techniques by which the representation of the members of a given alphabet may be changed such that they are represented by members of another alphabet, namely substitution and transposition.

1.4.2.2 Substitution and transposition. All ciphers employ either or both of these techniques for the encryption of data. Substitution is the technique of replacing the letters of one alphabet with the letters from another alphabet (or other alphabets), whereas transposition is the technique of re-ordering the letters of a given alphabet. For example, a cipher which translates the members of a given alphabet into other members of the same alphabet by means of an offset alphabetic position is known as a mono-alphabetic transposition cipher. Here the n^{th} letter of the alphabet is replaced by the $(n+d)^{th}$ letter of the same alphabet, that is :-

$$A(n) \rightarrow A(n+d) \qquad \ldots\ldots \; 1.1$$

for $1 \leq n \leq l$

where $A(n)$ = n^{th} member of the alphabet A

l = length of alphabet A

d = alphabetic displacement

Mono-alphabetic transposition ciphers of this type are known as Cæsar Ciphers. Cryptanalysis of such ciphers is achieved by counting the letter frequency of the ciphertext, which will be of the same letter frequency pattern as the language of the plaintext, but displaced. A particular elaboration of the Cæsar Cipher is achieved by assigning each letter of the given alphabet a number, and multiplying the number of the letter to be encrypted by a constant k, where $gcd(k,l) = 1$ and l = length of the given alphabet, and adding a constant d. The result is then reduced modulo l. Thus for the English alphabet, $l = 26$ and :-

$$A(n) \rightarrow A((n \times k + d)(\text{mod } 26)) \qquad \ldots\ldots \; 1.2$$

and $gcd(k,26) = 1$

for $1 \leq n \leq 26$

Cryptanalysis of such a cipher is achieved by counting the letter frequency of the ciphertext and, in comparison with the letter frequency of the plaintext, attempting a trial and error approach involving the solution of simultaneous equations of the form:-

$$n' = n \times k + d \qquad \ldots\ldots \; 1.3$$

$$\text{where } n' = ((n \times k + d)(\text{mod } l)) \qquad \ldots\ldots \; 1.4$$

An example of mono-alphabetic substitution cipher is one which translates the members of a given alphabet into other members of an alphabet which has the same members as the given alphabet, but which has a shuffled order. Thus the alphabetic displacement is dependent upon the letter concerned. Such a mono-alphabetic substitution cipher will, for the English language, have 26! ($\approx 4 \times 10^{26}$) possible alphabets, whereas the Cæsar Cipher has only 25 possible alphabets for the English language. Cryptanalysis of such a mono-alphabetic substitution cipher is achieved by counting the letter frequency of the plaintext, which will be of the same letter frequency as the language of the plaintext, but in a shuffled order.

A poly-alphabetic cipher translates the members of a given alphabet into other members of more than one alphabet. Since more than one alphabet is used, the members of a given alphabet cannot be merely transposed and thus poly-alphabetic ciphers employ substitution. An example of a poly-alphabetic substitution cipher is the Vigenère cipher, where a plaintext message is enciphered by means of a "tableau" consisting of the plaintext alphabet and all possible new alphabets generated by successive displacement by one character of the plaintext alphabet, as shown in Figure 1.8.

Plaintext data is enciphered by means of a key word, that determines which displaced alphabet will be used to substitute for each plaintext letter. For example, using the key "open", the plaintext message "mary had a little lamb" is enciphered as follows:-

Plaintext :	mary had a little lamb
Keyword :	open ope n openop enop
Ciphertext :	apvl vph n zxxgzt pnaq

The idea of using such a poly-alphabetic substitution cipher is to "hide" the distribution of the plaintext language by spreading it between the different ciphertext alphabets. A more thorough explanation of such ciphers and their cryptanalysis is given in Chapter Two.

1.4.2.3 Product ciphers. To further prevent successful cryptanalysis of a cipher, a development from using several ciphertext alphabets (that is, a poly-alphabetic cipher) is to use several ciphers to encrypt plaintext. This is known as a product cipher, where the encryption function E is a composite of t functions (ciphers) F_i, $1 \leq i \leq t$,where each F_i may be either a substitution or a transposition. Due to the intrinsic complexity of the manipulations involved in product ciphers, they need to be implemented using cipher machines. Early cipher machines such as the Jefferson Cylinder and the Wheatstone Disk were succeeded by rotor cipher machines such as the ENIGMA machine and the Hagelin machine.

All such rotor-based cipher machines employ symmetric algorithms; the same cryptographic key being used both for encrypting the plaintext and for decrypting the resulting ciphertext.

The most well known modern substitution-permutation product cipher is the Data Encryption Algorithm (or DEA), published in 1977 as part of the Data Encryption Standard (or DES) [9]. It is still very widely in use today and is generally considered to be a bench-mark with which to compare the performance of other symmetric cryptographic algorithms. For this reason the DES is considered in detail in Chapter Two.

	ABCDEFGHIJKLMNOPQRSTUVWXYZ
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
B	BCDEFGHIJKLMNOPQRSTUVWXYZA
C	CDEFGHIJKLMNOPQRSTUVWXYZAB
D	DEFGHIJKLMNOPQRSTUVWXYZABC
E	EFGHIJKLMNOPQRSTUVWXYZABCD
F	FGHIJKLMNOPQRSTUVWXYZABCDE
G	GHIJKLMNOPQRSTUVWXYZABCDEF
H	HIJKLMNOPQRSTUVWXYZABCDEFG
I	IJKLMNOPQRSTUVWXYZABCDEFGH
J	JKLMNOPQRSTUVWXYZABCDEFGHI
K	KLMNOPQRSTUVWXYZABCDEFGHIJ
L	LMNOPQRSTUVWXYZABCDEFGHIJK
M	MNOPQRSTUVWXYZABCDEFGHIJKL
N	NOPQRSTUVWXYZABCDEFGHIJKLM
O	OPQRSTUVWXYZABCDEFGHIJKLMN
P	PQRSTUVWXYZABCDEFGHIJKLMNO
Q	QRSTUVWXYZABCDEFGHIJKLMNOP
R	RSTUVWXYZABCDEFGHIJKLMNOPQ
S	STUVWXYZABCDEFGHIJKLMNOPQR
T	TUVWXYZABCDEFGHIJKLMNOPQRS
U	UVWXYZABCDEFGHIJKLMNOPQRST
V	VWXYZABCDEFGHIJKLMNOPQRSTU
W	WXYZABCDEFGHIJKLMNOPQRSTUV
X	XYZABCDEFGHIJKLMNOPQRSTUVW
Y	YZABCDEFGHIJKLMNOPQRSTUVWX
Z	ZABCDEFGHIJKLMNOPQRSTUVWXY

Figure 1.8 The Vigenère Tableau

1.4.2.4 Stream and block ciphers. All ciphers can be categorized as being either a stream cipher or a block cipher. These are considered below.

Stream ciphers are used either where a communications system requires a continuous (synchronous) data link to be maintained or where there is insufficient memory capacity to allow a block of data to be stored. A truly synchronous stream cipher encrypts a given plaintext character both according to the position of the character in the plaintext stream and to the particular cryptographic key. The ciphertext character generated for a given plaintext character by a self-synchronizing stream cipher will depend not only upon the encryption key and upon the position of the plaintext character in the plaintext stream, but it will also depend upon a particular number of preceding ciphertext characters and hence upon the whole of the preceding plaintext character stream. This may have important implications for the propagation of errors; refer to Section 1.6.4. Stream ciphers are dealt with in greater detail in Chapter Two.

Block ciphers process plaintext characters in blocks of defined size. The size of the blocks is dependent upon the particular algorithm, and each block is processed independently of other blocks. The parallel nature of the processing performed in block ciphers makes them computationally efficient and ideal for general file transfer. The choice of whether to cause the result of an encryption of a given plaintext block to be dependent upon the nature of preceding plaintext or ciphertext data (in addition to the nature of the plaintext data, its position in the block and the particular cryptographic key being used) will depend upon the importance of more secure encryption in relation to the importance of the avoidance of error propagation between data blocks. Refer to Chapter Two for a more thorough consideration of the issues regarding the use of block ciphers.

1.4.2.5 Symmetric and asymmetric algorithms. All ciphers employ either symmetric or asymmetric algorithms. Symmetric algorithms have the property that plaintext encrypted using such an algorithm can be recovered from the corresponding ciphertext using the same algorithm in conjunction with the same cryptographic key, thus :-

$$F_k^{-1}(F_k(m)) = m \quad \ldots\ldots\ldots\ldots \quad 1.5$$

where $F_k(m)$ is the encryption of the plaintext message m using function F, with cryptographic key k.

and $F_k^{-1}(m)$ is the corresponding decryption

Since the same cryptographic key is used for both encryption and decryption it needs to be kept secret to protect the plaintext data from recovery by undesired parties. This has important implications for the distribution of cryptographic keys; refer to Section 1.6.2.4. This form of cryptography, referred to as symmetric algorithm cryptography, conventional cryptography or secret (private) key cryptography, has been the most common form of cryptography employed to date due to its ease of implementation and computational efficiency.

All the cryptographic algorithm examples given thus far in this chapter have been of symmetric algorithms.

Asymmetric algorithms have the property that plaintext encrypted using such an algorithm is not recoverable from the ciphertext so generated by using the same key that was used to encrypt the plaintext, thus :-

$$F_k^{-1}(F_k(m)) \neq m \qquad \ldots\ldots 1.6$$

where $F_k(m)$ is the encryption of the plaintext message m using function F, with cryptographic key k.

and $F_k^{-1}(m)$ is the corresponding decryption

Since different cryptographic keys are used for encryption of the plaintext and decryption of the corresponding ciphertext, the secrecy requirement of cryptographic keys does not apply to both keys in a particular encryption key - decryption key pair; merely to either one of the keys. This form of cryptography, referred to as asymmetric algorithm cryptography or public key cryptography, is a relatively recent form of cryptography which has become adopted only in the last decade due to its intensive processing requirements in comparison with symmetric algorithm cryptography, refer to Diffie and Hellman [10].

Thus an asymmetric algorithm cryptosystem may be established by selecting a suitable encryption key and publishing it whilst calculating the corresponding secret decryption key. Public key cryptosystems are dealt with in detail in Chapter Three.

The properties of public key cryptosystems have important implications for the distribution of cryptographic keys; refer to Section 1.6.2.4.

1.4.3 Authentication techniques.

An important requirement in any communications system is the ability to authenticate the communicating parties. There are many ways in which authentication may be achieved, but all authentication schemes involve the party to be authenticated being in possession of certain authentication-related data of which other, non *bona fide*, parties are not in possession. The party performing the verification of an authentication need not necessarily be in possession of this authentication-related data; refer to zero knowledge schemes in Chapter Three. Two important authentication techniques are one-way functions and digital signature schemes.

1.4.3.1 One-way functions. A one-way function is, as its name implies, a function which affects data such that it is intractable to determine its input given its output, even though the function is deterministic. This may be simply achieved by mapping the input (plaintext) message space onto a smaller ciphertext space, although more complex schemes may employ separate data to directly affect the plaintext or to modify the properties of the function. An example of a simple one-way function is given in section 2.5.3.

A one-way function may be used to authenticate a party by being used to generate a verification vector from an authentication vector supplied by a verifying party. A simple authentication scheme could be implemented by the verifying party selecting an authentication vector (ideally at random) and sending this to the authenticating party. Both parties would then pass this data through a commonly-known one-way function, the result of which would be a particular verification vector. The authenticating party would then return its verification vector to the verifying party, where the two verification vectors would be compared. Identical verification vectors would mean positive verification of the authenticity of the authenticating party.

Note that such a trivial authentication scheme such as this would require that the common one-way function be kept secret. This is both difficult due to the requirement for sharing a common one-way function for each communicating party, and fraught with risk of the secret one-way function being recovered by an unauthorized party. More realistic authentication schemes may involve digital signature schemes.

1.4.3.2 Digital signature schemes. These are schemes in which the party to be authenticated signs, with corresponding verification vectors, the authentication vectors sent to it by a verifying party. This will typically involve the use of a one-way function, but will also require secret authentication data. This will typically be in the form of a secret cryptographic key. The authentication vectors will typically be random in nature, and any one-way functions will be common between groups of communicating parties, rather than common between each individual pair of verifying / authenticating parties. The one-way functions will not need to be updated if properly chosen, and will not need to be kept secret. This applies equally to any other functions used for the same purpose in place of true one-way functions.

A digital signature scheme may employ either symmetric or asymmetric algorithm cryptography. Asymmetric algorithms may be used to implement a public key cryptosystem, as outlined in Section 1.4.2.5. However, instead of selecting a public encryption key and calculating a corresponding secret decryption key, the opposite is performed; a public decryption key is selected and the corresponding secret encryption key is calculated. Thus a digital signature scheme may be implemented using public key cryptography as follows. A verifying party sends a random authentication vector to the authenticating party, which generates a digital signature by passing the authentication vector through an encryption algorithm employing a secret encryption key. The resulting signature, or verification vector, is sent to the verifying party which passes it through a decryption algorithm employing a public decryption key. The recovered data is compared to the original authentication vector; identical authentication vectors would mean positive verification of the authenticity of the authenticating party. In this manner any party would be able to verify, using a public decryption key, the signature generated by an authenticating party using an authentication vector sent by the verifying party, but only *bona fide* authenticating parties would be able to generate the digital signature using the secret encryption key. Such an authentication exchange is depicted in Figure 1.9.

Symmetric algorithms may also be used to implement a digital signature scheme, but only where the verifying party is a trusted entity. In such a scheme the one-way functions (employing asymmetric algorithms) are replaced by symmetric algorithms with secret keys. Thus a typical authentication exchange involving symmetric algorithms may be as follows.

The verifying party selects an authentication vector, ideally a random number, and sends it to the authenticating party. Both the verifying party and the authenticating party encrypt the authentication vector using the common symmetric algorithm with a common secret encryption key (authentication key), previously known to both parties. The authenticating party sends the result of this, the verification vector or digital signature, back to the verifying party where it is compared to the locally generated verification vector. Identical authentication vectors would be considered as a positive verification of the authenticity of the authenticating party. Thus it can be seen that such an authentication scheme is confined to trusted parties only; secret authentication cryptographic keys also need to be known by the verifying party. Such an authentication exchange is depicted in Figure 1.10.

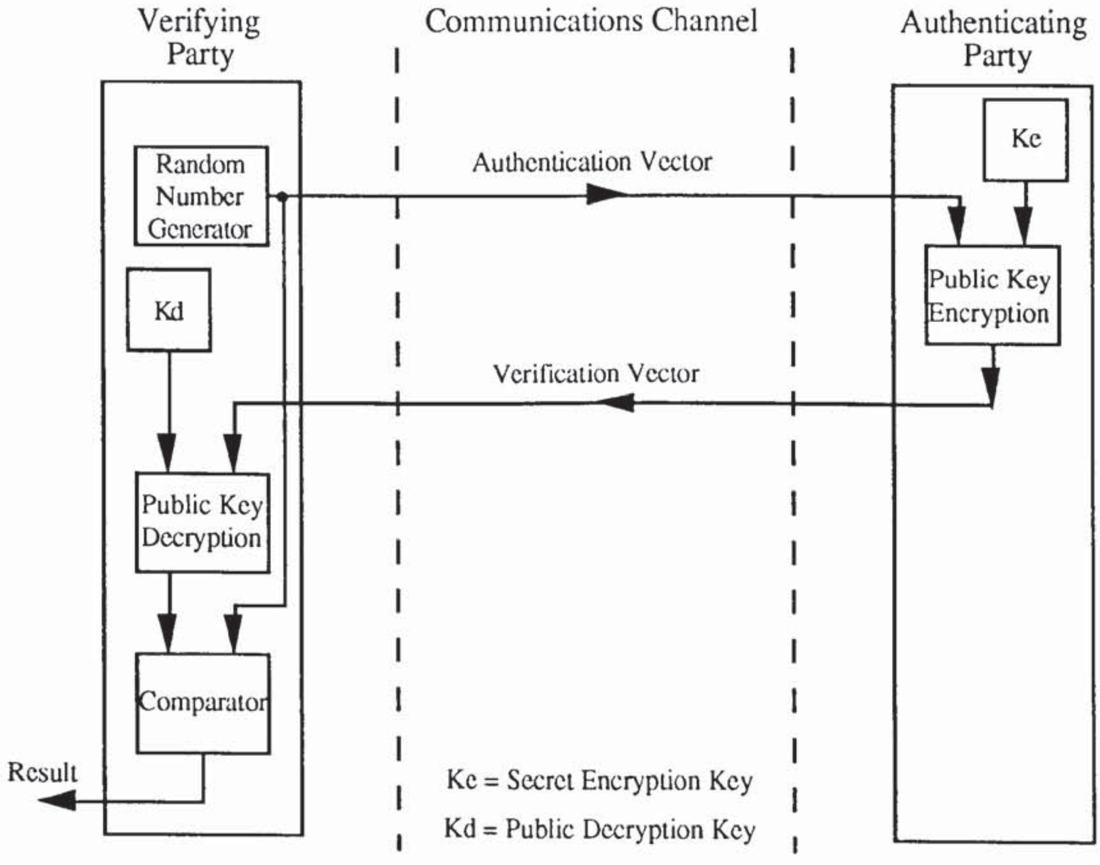


Figure 1.9 Authentication Using Asymmetric Algorithms

Note that an authentication scheme such as this would typically have a reduced processing overhead compared to the previous example employing public key cryptography, but it would be susceptible to an attack whereby successive verification vectors are used in an attempt to determine the secret encryption key; the encryption algorithm being both of a symmetric nature and presumably public knowledge.

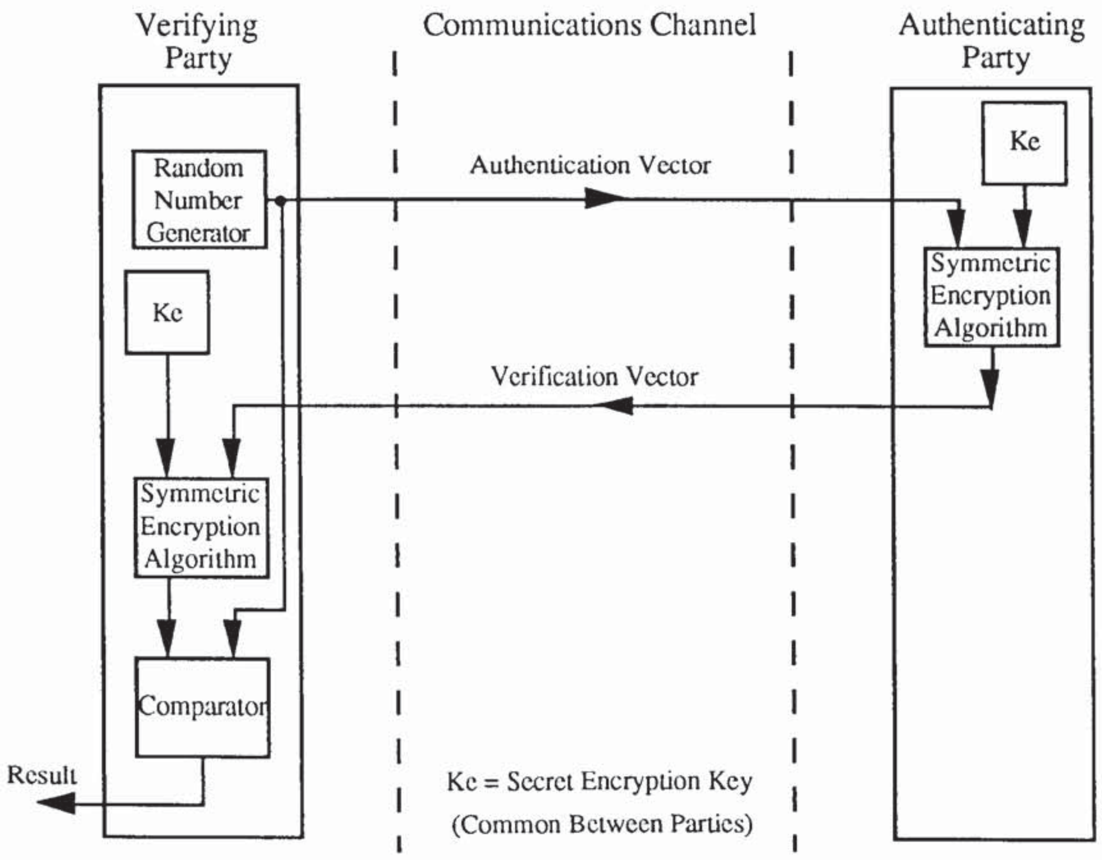


Figure 1.10 Authentication Exchange Using Symmetric Algorithms

1.4.4 Traffic intelligence and randomizing.

Another aspect of the security of digital communication systems is that of traffic intelligence. An attacker of a secure communications system need not necessarily be able to decipher the communicated messages; intelligence can be obtained merely from whether a particular party is communicating, when they are communicating and for how long they are communicating. This form of information, or traffic intelligence, can be of use in an attack in a tactical situation or where traffic analysis is being concentrated upon that of a particular party.

If combat of this form of attack is desired, the data communicated to or from a given party needs to be protected such that it is impossible to distinguish the data of this particular party from the data of other parties or possibly from both the data of other parties and from periods of communication inactivity. The former requirement may be satisfied over a channel providing for many parties, by multiplexing their data such that it is impossible for an attacker to distinguish the data of one party from any other party. The latter requirement would be satisfied, particularly over a channel providing for few parties, by the additional inclusion of false data to "fill in" for the periods of non-communication of real data.

The only way in which one message may be "hidden" in other messages and / or have periods of communication inactivity masked by the inclusion of false data, is when both real communications traffic and any false data appear random in nature. Thus it would be impossible for an attacker to distinguish one item of apparently random data from any other item of apparently random data.

The inclusion of apparently random data to mask periods of non-communication of real data is not a particular problem, since apparently random data may be readily generated. The crucial condition to be satisfied is that any real data, generated as a result of the encryption of party communications traffic, appears to be of a random nature to the cryptanalyst. The statistical analysis of random data for the purposes of cryptanalysis will yield no information of any use, since the distribution of all data items will be equal. Thus any cipher must transform the plaintext input to it into ciphertext of an apparently random nature. Note that since the output ciphertext of any particular cipher algorithm depends upon the plaintext input and upon the particular cryptovariable used, the output must be deterministic and hence cannot be truly random in nature. This does not, however, preclude a good cipher-cryptovariable combination producing ciphertext which, irrespective of the nature of the input plaintext, appears to be of a random nature and thus thwarts statistical cryptanalysis.

1.4.5 Hybrid schemes.

As has been mentioned in Section 1.4.2.5, the processing requirements of public key cryptography is considerably greater than that of private key cryptography. The use of public key cryptography to encrypt the bulk of communications traffic would, therefore, be process intensive compared to the use of private key cryptography. The properties of public key cryptography, however, make it ideal for the authentication of communicating parties and of data, and for cryptographic key distribution; the latter being a particular problem in the use of private key cryptography. A solution of the problems associated with the two forms of cryptography is to create a hybrid cryptosystem.

In such a hybrid cryptosystem, the asymmetric algorithms associated with public key cryptography are used to distribute cryptographic keys between communicating parties. This process is process intensive, but is only required for the initial establishment of a common secret cryptographic key between communicating parties. This common secret key would then be used by all communicating parties for conventional symmetric algorithm cryptography to encrypt and decrypt the bulk of the communications traffic at a much reduced processing overhead. Thus it can be seen that a hybrid cryptosystem can optimize the intrinsic properties of the two forms of cryptography.

1.4.6 Exploitation of digital hardware weaknesses.

As is outlined in Section 1.4.3.2, there are certain hardware weaknesses intrinsic in certain secure communications equipment which may be exploited by an attacker. As with analogue communications systems, digital systems may also suffer from unwanted coupling of the plaintext signal into the ciphertext bit stream or ancillary channels such as power lines. There may also be electromagnetic radiation which bears information of use to an attacker.

With digital systems, the manner in which a plaintext signal may couple into the ciphertext bit stream may be by simple superposition of the two signals or the plaintext signal may modulate the ciphertext bit stream. The mode of modulation could include pulse amplitude modulation, pulse position modulation or pulse width modulation.

The various potential weaknesses in the hardware associated with digital secure communication systems can only be overcome by careful physical design of such hardware.

1.5 Data Integrity Schemes.

Another area of security concerning the communication of data between parties is that of data integrity. The data which is sent from one party to another needs to be protected not only from the possibility of recovery by an attacker, but also from the possibility of modification by an attacker; such modification may be the substitution, transposition, erasure or addition of data. There is also a possibility of an attack by replay of messages that have already been transmitted, such an attack being generally referred to as a protocol reflection attack. Cryptosystems may not necessarily be susceptible to modification or replay attacks, but those which are can be protected by cryptographic checksums and time stamping, respectively.

1.5.1 Cryptographic checksums.

A cryptographic checksum is an item of data which is included in a given block of data to protect that block of data from any modifications being unnoticed, it being calculated from some or (ideally) all of the block of data. A cryptographic checksum may be calculated in several ways, of which two popular techniques are as follows.

The data to be protected is passed through an encryption function (be it symmetric or asymmetric) such that its output is determined by all the protected data. The encrypted data may then be reduced by hashing it such that a particular data field is mapped onto a smaller data field, refer to Rompel [11]. Refer to Section 2.5.3 for an explanation of cryptographic hash functions. This data then constitutes cryptographic check data, which may be appended to the original data. Any subsequent modification to the protected data will not result in a corresponding change in the cryptographic check data.

Another technique of protecting data with cryptographic checksums is to directly reduce the protected data using a one-way hash function, the result being appended to the protected data as above. In both cases, the aim is to generate check data which is determined by the data to be protected, but cannot be generated without knowledge of the particular cryptographic key or particular hash function being used.

1.5.2 Time stamping.

In order to avoid the possibility of an attack by replay of previous messages, a particular cryptosystem may employ time stamping. This is a technique of labelling data to be protected with other unique data which represents either a serial number or some chronologically-based number. The aim is to append such protective data to the data to be protected in order that each item of protective data is used only once; if an attempt is made to replay the message the protective data item will be invalid and hence the illegal message can be detected and rejected. Any protective time stamp data appended to a message would itself have to be protected by overall hashing or encryption of the combined message and time stamp, or by any time stamp data being included in the calculation of any cryptographic check data.

1.6 Cryptosystem Issues.

In all cryptosystems there are several generic aspects which need to be considered when selecting or designing such a system. These generic aspects, which often need to be evaluated with respect to each other, are broadly considered below.

1.6.1 Security level.

The security level afforded by a particular cryptosystem is paramount in its design considerations, since the basic objective in implementing such a cryptosystem is to protect access to the communication system facilities and access to the data contained in and communicated by the communication system. The security level can be considered in two broad categories; authentication and encryption.

1.6.1.1 Authentication. The security level provided by an authentication system can be considered as the probability of a successful impersonation attack upon the system. A popular form of authentication is a cryptographic challenge-response protocol; in such a protocol the authentication security level is the probability of an attacker successfully providing the required cryptographic response upon being provided with a (random) number and being challenged to respond. The probability of this occurring is 2^{-n} for an n-bit binary word, given two provisos: That the attacker has no knowledge of how to successfully generate the response, thus leaving only a random choice of response, and that the attacker has only one chance of successfully authenticating himself by providing the required response. Note that the number of attempts at authentication that are permitted has a direct bearing upon the security level, since repeated authentication attempts could be used in a manipulative statistical attack upon the cryptosystem.

For comparison, an authentication system requiring a simple n-bit password would have an initial security level of 2^{-n} given the first proviso above, but this would be reduced to $(2^n - x)^{-1}$ where x is the number of different attempts to find the password. This is a classical brute-force attack upon a cryptosystem; the entire key space being progressively searched by the attacker.

Thus it is shown that the protocols associated with cryptosystems are just as important as the cryptographic algorithms they employ, with regard to determining the overall system security level. It is obvious that the required authentication security level depends upon the threat of attack on the system and upon the desired cover time. The security level may be expressed in the form of a time-complexity function. Hellman [12] has suggested choosing 10^{40} as an absolute safe bound.

1.6.1.2 Encryption. The cryptographic security level provided by an encryption system can be considered as the probability of a successful attack on the system such that plaintext data can be recovered from the ciphertext data. The probability of such a successful attack depends upon three aspects; the statistical distribution of the ciphertext data, the quantity of available ciphertext data, and the category of attack able to be mounted upon the cryptosystem.

An ideal cryptographic algorithm will transform plaintext data into ciphertext data which is random in nature, having an even statistical distribution of characters within the ciphertext. Any realistic cryptosystem will create ciphertext data in a deterministic manner, such that it is dependent upon the corresponding plaintext and upon the particular cryptographic key being used. Thus the ciphertext data cannot be truly random in nature. A good encryption algorithm will, nevertheless, produce ciphertext data which is approximately random in nature. Any unevenness in the distribution of the ciphertext data may provide the cryptanalyst with useful information with a view to successful cryptanalysis of the ciphertext data.

The quantity of ciphertext data available to the attacker for cryptanalysis also has a direct bearing on the probability of a successful attack on the ciphertext data, since the probability of successful statistical cryptanalysis increases with an increase in the quantity of ciphertext data available for analysis. For this reason cryptographic keys should be changed frequently, ideally upon every new message. Cryptographic algorithms should have the property that security of the ciphertext depends purely upon knowledge of the particular cryptographic key and not upon knowledge of the cryptographic algorithm. Cryptographic algorithms should also have the property that a change in the cryptographic key produces a maximal change in the ciphertext data for given plaintext data. It is obvious that the required encryption security level depends upon the threat of attack upon the system and upon the desired cover time, but a minimum value for modern cryptosystems is 2^{-64}, (for a 64-bit key).

The category of attack able to be mounted on the cryptosystem is a prime consideration in determining the probability of a successful cryptanalytic attack on the ciphertext data, as outlined in Section 1.2.3. In order to thwart any possible chosen-plaintext attack the encryption algorithm should have the properties described above and any cryptographic keys used should be changed frequently.

1.6.2 Cryptographic key management.

Key management is a fundamental security requirement in all cryptosystems. The security level afforded by a particular cryptographic algorithm intrinsically depends upon the nature of the cryptographic key used, which may be required to be protected from being determined by or modified by an attacker. Key management can be subdivided into several aspects, four of which are considered below.

1.6.2.1 Random number generation. This is a fundamental requirement in all cryptosystems, since cryptographic keys should ideally be random in nature. A random choice of cryptographic key means that any knowledge of previous cryptographic keys will be of no use in determining the cryptographic key currently in use. Random numbers are also of particular use in cryptographic challenge-response authentication protocols. Truly random numbers are difficult to generate (and check), although pseudo-random numbers may well suffice. Such numbers can be generated either by digital or by analogue means.

1.6.2.2 Prime number generation. Prime number generation is a fundamental requirement in public key (asymmetric algorithm) cryptosystems, as outlined in Chapter Two. The generation of large prime numbers is a process-intensive task. Typical primes used in public key cryptosystems are in the region of 300 bits to 500 bits in length.

1.6.2.3 Elimination of weak keys. Certain cryptographic algorithms are susceptible to reduced security level due to weak cryptographic keys. These are keys which, due to the nature of the particular algorithm, provide less statistical security than other keys. Such weak keys can exist in both private and public key cryptography and should be eliminated from any key table by careful filtering of a (pseudo) random source. An list of weak keys for the DEA are given in the American National Standard [13]. The undue repetition of previously used keys is intrinsically weak and should be prevented by similar filtering.

1.6.2.4 Key distribution. The physical distribution of cryptographic keys between the required nodes in a particular network is an important consideration, particularly where the cryptosystem employs private key (symmetric algorithm) cryptography. Cryptographic keys may need to be distributed in a secure manner such that they can neither be recovered by an attacker nor can any modification by an attacker go unnoticed. Such requirements may be addressed by a key management scheme incorporating such aspects as hierarchical encryption, data integrity schemes and public key cryptography. Hierarchical encryption is a technique of implementing a tree structure of cryptographic keys such that, for example, an individual communication would be protected with a message key which could itself be protected by a session key which could in turn be protected with a master key.

1.6.3 Cryptosynchronization.

This is the requirement that plaintext data which was encrypted with regard to the particular contents of a register containing, say, data derived from previous plaintext data, is decrypted with regard to the same register data. This means that the encryption and decryption of data must be synchronized with respect to the ciphertext data; if data is lost from a ciphertext data file then this will result in the incorrect decryption of this data either from the point of occurrence of the ciphertext error and thereafter (for a stream cipher), or for the data block concerned (for a block cipher). The maintainability of cryptosynchronization is not generally a problem for digital systems, since they are inherently synchronous and will typically be protected from any data loss by channel coding.

1.6.4 Error propagation.

This is an important consideration in any cryptosystem; any encryption algorithm which employs cipher feedback or, for that matter, any block cipher, will be prone to error propagation to a certain extent. Communication systems which use channels prone to errors in transmission are particularly susceptible to any effects of error propagation, since frequent losses of data over a poor channel may result in a frequently increased loss of recovered data due to error propagation. This is particularly the case in mobile radio communication systems. Any communication system employing real-time communications will suffer a loss in real-time communications efficiency due to the effects of error propagation.

Error propagation may be controlled in cryptosystems employing block ciphers by limiting the size of data blocks. Robust channel coding schemes for the detection and correction of errors are essential in any mobile radio communication system.

1.6.5 Bandwidth preservation and data expansion.

Any communication system utilizing the air interface as a link between nodes is necessarily limited with regard to the bandwidth available for each channel. This is particularly the case for mobile radio communication systems, where bandwidth is strictly limited. In the latest digital mobile communication systems, such as the GSM900/DCS1800 and the DECT, the required channel bandwidth is reduced by employing source coding. The resulting digital coded data is channel coded before being transmitted via the air interface. This causes certain expansion of the transmitted data which, if transmitted at a given source information rate, results in a corresponding increase in the required channel bandwidth. It is thus important that any cryptosystem does not further increase the required channel bandwidth. This can be achieved by ensuring that any algorithms employed map plaintext onto a ciphertext area that is no larger than the plaintext area.

Such communication bandwidth requirement is not to be confused with any necessary signalling overhead; such an overhead will already exist in any multi-user mobile communication system in order to administer such aspects as call set-up, channel allocation and routing. This signalling overhead is bound to be increased by the inclusion of any realistic cryptosystem, but such an increase need only be dealt with upon such events as call set-up and cell handover.

1.6.6 Processing power / processing delay trade-off.

Any communications system / cryptosystem implementation is bound to be the result of several compromises between the various generic aspects inherent in such systems. For example, higher security levels demand increased signalling (due to longer cryptographic keys) but, more importantly, also demand increased processing. A particular amount of processing requires a particular amount of processing time with a particular level of processing power. The trade-off between processing power and processing delay is an important consideration, since increased processing power means higher cost and higher power requirement, and increased processing delay means more time delay between data input to a particular algorithm and its data output. The latter is of particular importance in real-time communications such as duplex speech communications.

1.7 Conclusions.

This chapter has provided a background to communications security. The reasons why an attack may be mounted on a communications system have been summarized and the different nature of attacks upon secure communication systems have been categorized.

It has been shown that, in order to assess the security requirements of a particular communications system, it is first necessary to assess the potential threat of attack upon the system concerned. It has been shown how the threat of attack may be considered in terms of how such a threat can be affected by the nature of the transmitted data, the vulnerability of the target and the available means of attacking the system. The latter has been shown to include brute force attack, masquerade attack and statistical attack. Such statistical attacks are divided into three categories, namely ciphertext only, known plaintext and chosen plaintext. Given an assessment of the threat of attack upon a particular communication system and the desired cover time, the security level required may then be established.

Security in analogue communication systems has been shown to be susceptible to casual eavesdropping. Analogue security techniques have been considered in the form of scrambling and noise masking. Analogue scrambling techniques have been considered in three categories; frequency domain, time domain and two-dimensional, of which two-dimensional scramblers have been considered to be the most secure. Digital implementation of analogue scrambling has been shown to be the most realistic method of physically implementing a flexible and secure analogue scrambling system.

The concept of noise masking has been shown to be a possible technique of supplementing the security of analogue communication systems.

The security weaknesses inherent in analogue scramblers have been considered in the categories of intuitive weakness and hardware weakness. Various techniques of attacking analogue scrambling systems by virtue of their intrinsic weaknesses have been considered, as have the problems of distortion and delay which are associated with such systems.

Security in digital communication systems has been shown to be less susceptible to casual eavesdropping than it is in analogue systems. It has been clearly demonstrated how digital communications are ideally suited to the incorporation of encryption techniques, by virtue of their generic nature, such digital encryption techniques being more versatile and more secure than analogue scrambling techniques.

An overview of cipher generic types has been presented, including alphabetic substitution and transposition, product ciphers, and block and stream ciphers. The various properties of these generic ciphers have been broadly considered with a view to their incorporation in mobile radio communication systems.

The requirements of party authentication schemes have been considered, with regard to an introduction to public key cryptography and to authentication techniques. It has been shown how the use of public key techniques for party authentication and key distribution and the use of private key techniques for bulk traffic encryption constitute a hybrid cryptographic scheme, which may be an optimum solution.

Possible schemes for ensuring the integrity of transmitted digital data have been introduced and various possible hardware weaknesses, which could be exploited by an attacker, have been outlined.

The various generic properties of cryptosystems have been broadly considered, in conjunction with the generic properties of communication systems, with regard to their impact upon cryptosystem design. Such properties include the security level (of traffic encryption and of party authentication), cryptographic key management, cryptosynchronization, error propagation, data bandwidth preservation, and processing power and delay.

This chapter has presented an overview of analogue scrambling schemes and digital encryption schemes, with regard to the properties of communication systems and to the threat of attack upon such systems. It has been shown how digital encryption systems can offer the system designer very high security levels for both party authentication and for traffic encryption, in addition to offering the option of extra security-related facilities such as cryptographic key distribution and data integrity schemes. The essential properties offered by digital cryptosystems are complete flexibility in their implementation and quantitative levels of security. This, in conjunction with their intrinsic compatibility with the digital nature of the Second-Generation of PLMN systems, means that the use of digital cryptographic techniques is by far the best means of implementing security in such digital communication systems.

CHAPTER 2. SYMMETRIC ALGORITHM CIPHERS AND CRYPTOSYSTEMS.

This chapter presents an overview of the theory pertinent to secrecy systems and to cryptosystems. Perfect secrecy is investigated in relation to cryptosecrecy, and the latter is examined with reference both to work factor and to cover time. The general authentication security requirements are considered in relation to the security mechanisms required to implement them.

The particular characteristics and uses of symmetric algorithm (private key) cryptography are examined, with particular emphasis on block ciphers and pseudo-random key streams. The DEA is used as a particular example and its modes of operation are analysed in detail. The modes of operation of block ciphers are examined with regard to the type of application. Such types of application examined include ciphering and cryptographic key distribution.

2.1 Encryption Security Requirements.

All cryptosystems have to be definable with regard to the level of security they afford. This (quantitative) level of security applies to whatever type of security is being provided by a given cryptosystem. It is necessary that the general theory pertaining to the secrecy level afforded by cryptosystems be considered as a precursor to the consideration of the secrecy level of any specific cryptosystem. There are two fundamental types of security that a cryptosystem can provide; data encryption and (party) authentication.The security afforded by a data encryption system is classically defined by Shannon [14].

2.1.1 Shannon's theory of secrecy systems.

Cryptography aims to protect message data by representing the message data by strings of purely random data, thus thwarting any attempt of cryptanalytic attack. Such protection of data representing a message M can be divided into three essential categories :-

- The prevention of successful cryptanalysis of the message M
- The prevention of the ability to change the message M to M' such that M' is accepted by the receiving party as M.
- The prevention of the ability to initiate messages, such that the messages are accepted by the receiving party as being *bona-fide* messages from another (initiating), party.

The first security problem, generally referred to as the privacy problem, is the most well known of the security categories. The importance of the second and third security categories, (the authentication problem and the non-repudiation problem, respectively), has increased with the general growth of public access communication systems.

The security of a message, with regard to the amount of privacy provided, is related to the difficulty in successfully inverting the encryption transformations which originally created the ciphertext from the corresponding plaintext message. Shannon [14] defined a system with perfect security where if an attacker knows E (the encryption transformation), and is in possession of an unlimited amount of ciphertext, any attempt to elicit the original message from the ciphertext will result in a choice having to be made from all possible messages within that message space. Thus perfect security can be defined as :-

$$P_C(M) = P(M) \quad \dots\dots \quad 2.1$$

where	C	=	$E(M)$
and	M	=	message
	C	=	ciphertext
	$P(M)$	=	probability that M will occur
	$P_C(M)$	=	probability that M was sent given that C was received

Then :-

$$P_M(C) = \sum_{K, E_K(M) = C}^{\mathbf{K}} P(K) \quad \dots\dots \quad 2.2$$

where	$P_M(C)$	=	probability that C was received given that M was sent
and	$P(K)$	=	probability of keys that transform M into C
	$\mathbf{K}$	=	across key space of K

Thus $P_M(C)$ is the sum of the probabilities $P(K)$ of the keys that encrypt M as C. Usually there will only be one key which satisfies $E_K(M) = C$. Thus for perfect secrecy :-

$$P_M(C) = P(C) \quad \dots\dots \quad 2.3$$

for all C

Thus the probability of receiving ciphertext C is independent of the message M. Perfect secrecy can only be assured when the length of the key is the same as the size of the message; that the key space is the same as the message space. This maximizes the uncertainty of an attacker in an attempt to elicit the key and message. The only such system is the one-time pad, where the key is chosen at random and is discarded after each message. An automated version of the one-time pad is the Vernam Cipher, refer to Davies and Price [15]. That the attacker possesses any amount of ciphertext C makes no improvement in the probability of successfully recovering the message M. The message M is recovered by evaluating D_K (C), where D_K is the corresponding decryption transformation to the encryption transformation E_K, using key K.

Such systems that are based on Shannon's equivocation are unconditionally secure and will resist unlimited cryptanalysis. The security of such systems is derived purely from statistical uncertainty. Note that the key space is equal to the message space in a Vernam cipher.

2.1.1.1 Entropy. A cipher is considered to be unconditionally secure if the entropy of its key K never approaches zero for any length of message M. The amount of information in a message is measured by its entropy, refer to Gibson *et al* [16]. If $M_1, M_2, \ldots, M_n$ are n possible messages and $P(M_1), P(M_2), \ldots, P(M_n)$ are their respective probabilities of occurring, then the entropy H (M) of the message is defined as :-

$$H(M) = \sum_{M} P(M_i) \log_2 \left[\frac{1}{P(M_i)} \right] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.4$$

Each $\log_2(1/P(M_i))$ term represents the number of bits needed to encode the message in an optimal manner.

When all messages are equally likely, that is $P(M_1) = P(M_2) = \ldots = P(M_n) = 1/n$, then $H(M) = \log_2 n$. If n = 2k, then k bits are needed to encode the message, where n is the possible number of messages. The value of H (M) ranges between a maximum of $\log_2 n$ and a minimum of 0 (when n = 1 and P (M) = 1). The entropy of a message can be considered to be its uncertainty; the number of bits required for the recovery of a message concealed by cryptographic transformations. For example, the maximum entropy of a random binary digit is attained with the uniform distribution $P(M_1) = P(M_2) = 0.5$, where :-

$$H(M) = \log_2 2 = 1 \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.5$$

2.1.1.2 Rate of language and redundancy. The rate r of a particular language, for messages of length k, is given by :-

$$r = \frac{H(X)}{k} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.6$$

The rate of language denotes the average number of bits of information in each character; refer to Bluhme [17]. The absolute rate of a language is the maximum number of bits of information that could be encoded in each character assuming that all combinations of characters are equally likely. The absolute rate of language R is given by :-

$$R = \log_2 k \qquad \text{.......} \quad 2.7$$

This is the maximum entropy of the individual characters. For the English 26-letter alphabet, R = 4.7 bits/letter. The actual rate of English is much less (R = 3.2 bits per letter). This is because English, like all natural languages, is highly redundant. The redundancy inherent in the English language is due to the natural probability distribution of letters (or letter frequency) and of di-grams and tri-grams. The redundancy D of a language is defined in [17] as :-

$$D = R - r \qquad \text{.......} \quad 2.8$$

For the English language, with r = 1 and R = 4.7, the ratio D/R gives the redundancy factor, which is approximately 79 %. Thus the English language is only about 21% efficient.

Note that the more redundant a language is, the stronger the statistical relationship between the letters in a sequence. Conversely, if a language has no redundancy, then the occurrences of adjacent letters are statistically independent. To evaluate the entropy for an n-letter language H_n (M) requires an amount of calculation that grows exponentially as a function of n. The real redundancy of a language [17] can be expressed as :-

$$r_\infty = \lim_{n \to \infty} \left[\frac{H_n(M)}{n} \right] \qquad \text{.......} \quad 2.9$$

where n = number of letters in the message
and H_n (M) = entropy of message M

This value of redundancy can only be estimated.

2.1.1.3 Equivocation. This is defined as the conditional entropy of message M given that ciphertext C has been received; refer to Vantilburg and Boekee [18]. Thus the equivocation H_c (M) is given by equation 2.10, below :-

$$H_C(M) = \sum_{C} P(C) \sum_{M} P_C(M) \log_2 \left[\frac{1}{P_C(M)}\right] \quad \ldots\ldots\ldots\ldots\ldots\ 2.10$$

where $P_C(M)$ = conditional probability of message M given ciphertext C has occurred

Shannon [14] measures the secrecy of a cipher with respect to its key equivocation $H_C(K)$, for ciphertext C and key K. It may be expressed as the degree of uncertainty in K given C, and may be expressed thus :-

$$H_C(K) = \sum_{C} P(C) \sum_{K} P_C(K) \log_2 \left[\frac{1}{P_C(K)}\right] \quad \ldots\ldots\ldots\ldots\ldots\ 2.11$$

where $P_C(K)$ = probability of key K given that ciphertext C has occurred

If $H_C(K) = 0$, then there is no uncertainty in the cipher; it would then be breakable.

2.1.1.4 Unicity distance. The unicity distance of a message M is defined as the minimum message length that forces $H_C(M)$ to be approximately equal to zero. Similarly, the unicity distance of a cipher is defined as the minimum message length that forces $H_C(K)$ to be approximately equal to zero; refer to Damours *et al* [19]. Thus the unicity distance of a cipher is the amount of ciphertext needed to uniquely determine the key. As the amount of the ciphertext increases, the equivocation of the cipher decreases. Ciphers may be classified as being either of finite unicity distance, or of infinite unicity distance. The latter class of ciphers is referred to as *ideal* ciphers which are unbreakable.

An indication of the amount of ciphertext N needed to break the cipher is given by :-

$$N \simeq \frac{H(K)}{D} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 2.12$$

where D = redundancy of language
and $H(K)$ = information content of the key

2.1.2 Work factor and cover time.

Shannon's theory for perfectly secure systems assumes that any attacker has unlimited access to computing power. On the basis that all real computing systems are necessarily limited in their computing power, absolute secrecy is not required.

What is required is, rather than just an appreciation of the uncertainty and the unicity distance of a ciphertext message, a work factor to indicate the ratio of the amount of processing required to successfully cryptanalyse the particular cryptosystem. It is this computation requirement that is the essential property in assessing the strength of a particular cryptosystem. It allows the cryptosystem designer to assess how long it wold take a cryptanalyst to successfully cryptanalyse the cryptosystem, given a particular processing power. The security level of a particular cryptosystem can be expressed in terms of the number of person/computer years required to successfully cryptanalyse the cryptosystem. This leads to the concept of cover time.

Cover time is that amount of time during which an encrypted message is considered to be safe from successful cryptanalysis, given a particular level of processing power. Thus a cryptosystem designer will consider either the current processing power assumed to be available to a particular attacker, or the maximum processing power theoretically available to anyone; this choice depends upon whether the cryptosystem is to be immune from successful cryptanalysis either by a particular attacker, or by any possible attacker. The length of time a message is required to be secure is then used in conjunction with the assumed maximum processor power available, to determine the work factor required to secure the message. Note that the constant increase in available processor power as the required cover time expires will need to be taken into account in determining the required work factor.

2.1.3 Perfect secrecy and cryptosecrecy.

Shannon's theoretical cryptosystems offer perfect secrecy. Real cryptosystems have to function within necessary bounds of processing power and processing time, and thus statistical security is given by means of a work factor. This leads to the concept of cryptosecrecy. Note that the level of difficulty in successfully cryptanalysing a cipher message is asymptotically defined for a perfectly secure cryptosystem, but for a cryptosystem that relies upon cryptosecrecy it relies upon the concept of intractability. There is, however, at present no algorithm that can prove that a cryptosystem is crypto-secure. The only method of establishing an indication of the level of security offered by a crypto-secure cryptosystem is by the various heuristic methods employed by cryptanalysts. This is called certification. This remains the approved and respected means of establishing the security of a real cryptosystem, even though the technique is inexact and not mathematically provable. Complexity theory may prove to be the solution to this problem.

2.2 Authentication Security Requirements.

An essential function in most cryptosystems is that of authentication. As mentioned in Chapter One, there are three categories of protection in providing data security; namely privacy, authentication and non-repudiation. These categories are distinct, however they are not necessarily mutually exclusive. This depends upon the nature of the cryptosystem. It is most common, however, to address the problems of message privacy and of authentication directly, and to use these security provisions to provide an inherent non-repudiation security. A consideration of equipment authentication is included here, since this is used in conjunction with user authentication in many public communication systems as a necessary device to help combat theft and fraud.

2.2.1 User authentication categories.

In order to authenticate a user (of a cryptosystem), the user must be uniquely identified. Note that, as is the case for all real cryptosystems, any "unique" identification will, in reality, be a probabilistic identification. There are several ways in which a user may be authenticated. These may be categorized as follows :-

- Personal physical characteristics.
- Personal physical possession.
- Personal data possession.

In other words, the authentication can be based on something one is, on something one has, or on something one knows, respectively. To base authentication on physical characteristics requires sophisticated hardware (and software), and the resulting data would still need to be protected cryptographically over a network. This limits authentication to the remaining two methods. To rely on a purely physical possession for authentication is unfeasible, since any characteristic of such a possession would have to be translated into a logical form, and would then be subject to the same data protection requirement as above. A second drawback with this type of authentication is that there is no secure way of relating the identity of the physical possession to the identity of the user. The use of personal data possession for user authentication is an excellent solution to the latter problem of secure relationship. It is not without its own problems, however. These include the first problem of secure transfer over a network, the amount of data needed to be remembered and supplied by the user upon authentication request, and the secure administration of such a scheme.

A popular solution to the problem of authentication is to combine the logical possession by a user with a physical possession of an intelligent token; an identity module. This allows the cryptosystem designer to combine personal logical possession with personal physical possession in a manner which allows (relatively) large amounts of data to be used for authentication, complex calculations to be performed by the authenticating party, and secure communications between the identity module and the verifying party, such that secret authentication data is not disclosed. The most common form of such an identity module is the smart card. Refer to Chapter Six for a detailed consideration of smart cards and their protocols. Authentication protocols in general are considered in Chapter Four.

2.2.2 User authentication requirements.

There are several requirements regarding the authentication of users in a particular (communication) system, all or some of which may be implemented using cryptographic techniques. These authentication requirements include the following :-

2.2.2.1 The verification of the authenticity of the identifying party. This is a fundamental authentication requirement. If any party is to have a service provided to it by a system, then the party must be proved to be *bona fide*. Various authentication schemes for the Second-Generation PLMN systems are considered in Chapter Five.

2.2.2.2 The verification of the authenticity of the service provider. This is an important supplement to system authentication security; an attacker may choose to impersonate a service provider, refer to Chapter Five for service provider authentication and to Section 5.3.3.5 for an example of the above type of attack.

2.2.2.3 Verification of authenticity in a manner that is secure from third party impersonation. This is an essential aspect in any mobile communication system, due to the use of the air interface. Any third party may be privy to the communications over the air interface, and it is thus imperative that such communications do not compromise the system authentication security by allowing an attacker to use the transmitted data to impersonate a *bona fide* user. Examples of such secure authentication schemes are given by Königs [20].

2.2.2.4 Verification of authenticity in a manner which does not require a trusted verifying party. A development of the above secure authentication schemes is provided for verification of user authenticity by a verifying party which is not trusted to hold secret authentication/verification cryptographic data. Details of such schemes are given in Chapter Three.

2.2.3 Equipment authentication requirements.

As part of a secure environment for communications, it may be required to implement equipment authentication. Such equipment authentication may enhance user authentication, ensure equipment compatibility and provide protection against theft and fraud. Equipment authentication can be applied both to user equipment by the service provider, and to the service provider's equipment by the user.

2.3 Symmetric Algorithms and Private Key Cryptography

As explained in Chapter One, symmetric algorithms are conventionally used to encrypt plaintext into ciphertext for secure storage and transmission, and to decrypt the ciphertext to recover the original plaintext. Due to the fact that the cryptographic keys used for both encryption and decryption are identical, the keys used have to be commonly known by both the encrypting party and the decrypting party. Thus the keys are secret (private). The secure distribution of such keys is considered in Section 2.5.4.

2.3.1 Properties of symmetric algorithms.

Symmetric cryptographic algorithms must have two essential properties; they must allow the recovery of plaintext from ciphertext using the same cryptographic key that was used to originally generate the ciphertext from the plaintext, and the ciphertext generated must be of such a nature that it is an intractable problem either to recover the plaintext from the ciphertext without the particular cryptographic key, or to recover the cryptographic key from the plaintext. Note that the latter property of intractability is due to the randomizing effect of a cipher on plaintext, and is an essential property of any type of cryptographic algorithm and its key.

2.3.1.1 Distribution and cryptanalysis. All cryptographic algorithms act on input plaintext so as to modify the distribution of character frequencies within the ciphertext alphabet. An ideal distribution is flat, such that the probability of occurrence of any particular character in the ciphertext alphabet is the same as all others; the occurrence should be at random and be independent of the character distribution in the plaintext alphabet. If the occurrence is not at random, then a statistical cryptanalysis may be attempted with a finite probability of successfully recovering the key (cryptovariable) used, and the plaintext corresponding to the ciphertext.

An equivocation of cryptographic systems is shown in Section 2.1.1.3.

2.3.1.2 Error propagation. An important property of any cryptographic algorithm is the effect of errors occurring in the ciphertext, upon the successful recovery of the corresponding plaintext. The effect that an uncorrected error will have on plaintext recovery is dependent upon whether there is any form of cipher feedback employed by the particular cryptographic algorithm.

Cipher feedback is employed by some cryptographic algorithms to improve the distribution of the characters in the ciphertext alphabet. The result of such feedback is that the cryptographic transformation of a particular plaintext character, upon encryption, into its corresponding ciphertext character, will not just depend upon its value and that of the particular cryptovariable used, but also upon the value of preceding plaintext or preceding ciphertext. It is this dependency upon preceding plaintext or upon preceding ciphertext, that can permit errors occurring in the ciphertext to cause error propagation in the recovered plaintext.

To combat this effect, but still retain the advantage of cipher feedback, the use of a block cipher can restrict any error propagation to a limited number of blocks, or even to just one block. Such limitation of error propagation is dependent upon the nature of the cipher feedback employed.

An example of how such cipher feedback may be employed is depicted later in Section 2.4.3.2; refer to Figure 2.1.

2.3.2 Uses of symmetric algorithms in private key cryptosystems.

Symmetric algorithms may be employed in one of two ways in order to achieve data encryption. Either the plaintext may be encrypted one block at a time by a block cipher, or the block cipher may be used to generate a pseudo-random stream of data, which may then be used to encrypt the plaintext as a stream cipher. The usual method of encrypting (and decrypting) data with a stream of random, or pseudo-random, data is by means of bit-wise XOR (that is, modulo-2 addition).

2.3.2.1 Pseudo-random key streams and the Vernam cipher. The principle of using a pseudo-random key stream is that a stream of data with good randomness properties may be used to encrypt or decrypt data in real time at great speed. Such a pseudo random stream of data, when used to transform a stream of plaintext into a stream of ciphertext, or *vice versa*, is known as a key stream, the length of which should be considerably greater than that of any message stream to be encrypted. The advantage of using a pseudo-random key stream is that the stream may be generated in advance of encryption or decryption, and need not be securely stored or transmitted as a truly random sequence would have to be. Such a pseudo-random key stream may be generated at both ends of a communications channel; there only being a shared secret key for the purposes of generating the key stream. Such generation may be by the use of a block cipher, such as Data Encryption Standard [9].

The Vernam cipher uses a truly random bit stream to encrypt a plaintext message. When a truly random bit stream is used to bit-wise encrypt a plaintext message, and the random stream is at least as long as the plaintext sequence and the random sequence is discarded after each occasion of use, then absolute secrecy is achieved within the resultant ciphertext. Such absolute secrecy is as postulated by Shannon, and is synonymous with the use of a one-time pad. These are the only systems achieving absolute secrecy, which may well be greater than is realistically required of a cryptosystem. Note that such systems have an ultimate reliance on a secure distribution and storage of the complete random sequences to be used.

2.3.2.2 Block ciphers and their modes of operation. As stated previously, block ciphers can operate in more than one mode. The simple mode of operation is electronic code book, or ECB, where an input plaintext block undergoes substitution and / or permutation about itself, dependent upon the nature of a particular cryptographic key. This mode of operation, and several others, are considered in Section 2.4.3, which uses the DEA as an example.

2.4 The Data Encryption Standard.

The Data Encryption Standard (DES) was released in 1977 by the National Bureau of Standards, for United States Government encryption of non-classified information. The latest update uses the American National Standard ANSI X3.92-1981 for data encryption; the idea being to promulgate a standard Data Encryption Algorithm (DEA) for worldwide adoption by commercial and financial organizations. Diffie and Hellman proposed a theoretical brute-force attack upon the algorithm, which resulted in its US government accreditation being withdrawn. It remains, nevertheless, the most popular civilian symmetric algorithm cryptosystem, and is heavily used in banking organizations. Such is the popularity of the DEA, that it is commonly considered to be the bench-mark for symmetric algorithm cryptosystems. There have been several proposals for improving the security of the DEA, based either upon increasing the length of the 56-bit key, or upon multiple (super) encryption. The algorithm is broadly considered below.

2.4.1 The Data Encryption Algorithm (DEA).

The Data Encryption Algorithm works by processing 64-bit blocks of data in successive register stages. An input data block undergoes an initial permutational transposition, such that $D_0 = IP(D)$, where IP is an initial permutation, as defined in fixed permutation tables; refer to Seberry and Pieprzyk [21]. D_0 consists of L_0 and R_0, which are the left and right halves of D_0, respectively. L_0 and R_0 are passed through function F_{k_x}, with a cryptovariable of K_1, resulting in L_1 and R_1, thus :-

$$(L_{x+1}, R_{x+1}) = F_{k_{x+1}}(L_x, R_x) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.13$$

Sixteen iterations of function F are performed. In this manner the Data Encryption Algorithm constitutes a product cipher, combining both substitution and transposition. Finally, the resulting data L_{15} and R_{15} are concatenated via an inverse initial permutation, such that $D' = IP^{-1}(D_{15})$, where $D_{15} = (L_{15}, R_{15})$. This forms a 64-bit output data block, D'.

The function F is utilized as follows :-

$$\text{If} \qquad T_x = (L_x, R_x) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.14$$

where T_x denotes the result of the x^th iteration
and $L_x = t_1 \ldots t_{32}$, $R_x = t_{33} \ldots t_{64}$

Then $L_x = R_{x-1}$ 2.15

and $R_x = L_{x-1} \oplus F_{k_x}(R_{x-1})$ 2.16

where K_x is a 48-bit key derived from an initial 64-bit key, as defined in [21].

The function F is defined as follows. The initial step is to expand R_{x-1} from a 32-bit block to a 48-bit block. This is achieved using an expansion table, as defined in [21]. The expansion is made by selecting some of the bits of R_{x-1} more than once. The second step is to bit-wise XOR the 48-bit expansion $E(R_{x-1})$ with K_x, the 48-bit key. That is, bit-wise addition (modulo-2). The result of this is divided into eight 6-bit blocks, $B_1...B_8$, where :-

$E(R_{x-1}) \oplus K_x = (B_1, B_2....B_8)$ 2.17

Each of the 6-bit blocks is then used as an input vector to a substitution function (S-box) S_i, which returns a 4-bit block $S_i(B_i)$. These blocks are concatenated with each other; the resulting 32-bit block being transposed by the permutation function, P, as defined in [21]. Thus the overall effect of $F_{k_x}(R_{x-1})$ is :-

$P\ (\ S_1(B_1),....,S_8(B_8)\ )$

Each S-box, S_i, maps a 6-bit block $B_i = (b_1, b_2, b_3, b_4, b_5, b_6)$ into a 4-bit block as defined in [21].

Each iteration uses a different 48-bit key, K_x, derived from the initial key 64-bit key K. Note that the eighth bit of each 8-bit byte of the 64-bit initial key is a parity bit. These eight parity bits are discarded in the first permutation FP; the remaining 56 bits being transposed as defined in that permutation table. The result of the first permutation FP(K) is subsequently split into two halves C and D, which are used to derive each key K_x. If C_x and D_x denote the values of C and D used to derive K_x, then :-

$C_x = LS_x(C_{x-1})$ 2.18

$D_x = LS_x(D_{x-1})$ 2.19

where LS_x = number of left circular shift positions

and C_0, D_0 = The initial values of C and D, respectively.

The key K_x is then given by:-

$$K_x = SP(C_x, D_x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.20$$

where SP = The second permutation.

Decryption of data is achieved using the same algorithm and key, except that K_{16} is used in the first iteration, K_{15} in the second iteration, and so forth until K_1 is used in the sixteenth iteration. Note that the final permutation, IP^{-1}, is the inverse of the initial permutation, IP, and:-

$$R_{x-1} = L_x \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.21$$

$$L_{x-1} = R_x \oplus F_{k_x}(L_x) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.22$$

Note also, that after the last iteration, the left and right halves are not exchanged; instead the concatenated block $\{R_{16}L_{16}\}$ is input to the final permutation, IP^{-1}. This is necessary in order that the algorithm may be used both to encrypt and to decrypt. To this end, for decryption the order of the keys is reversed, but the algorithm itself is not.

2.4.2 Effects of the DEA on plaintext.

It can be seen from the nature of the iterative function that is applied to the blocks of plaintext, that the DEA performs successive rounds of block substitution upon the plaintext, in conjunction with initial and final stages of block transposition (permutation). The rationale behind this algorithm is to employ product cipher techniques, with iterative application of substitution, in a format such that the process may be readily implemented in a "pipeline" manner.

The successive rounds of substitution result in a progressive re-distribution of the frequencies of occurrence of the characters in the plaintext alphabet. This has a levelling effect which precludes statistical cryptanalysis of the resultant ciphertext. In other words, although the ciphertext output of the DEA is deterministic upon the nature of both the plaintext and the cryptovariable used, it will appear to be random in nature upon attempted cryptanalysis, irrespective of the nature of the plaintext. Note that the cryptovariable must be chosen with care, as explained in Section 1.6.2.3.

2.4.3 Modes of operation of the DEA.

The DEA has four modes of operation, as specified in the DES [9]. The four modes of operation are detailed in the following sections :-

2.4.3.1 Electronic Code Book (ECB). This is the fundamental mode of operation of the DEA, where 64-bit plaintext blocks are encrypted using a 56-bit key. This is the simple throughput mode, where 64-bit blocks of plaintext, P_i, are transformed into 64-bit blocks of ciphertext, C_i, or vice versa. The cryptovariable here is the cryptographic key used for the DEA, K_i. This mode of operation has the advantage of lack of error propagation (outside any given block), but data integrity cannot be guaranteed, since data blocks may be added, removed or modified without knowledge of the cryptovariable.

2.4.3.2 Cipher Block Chaining (CBC). This mode of DEA operation requires that each 64-bit ciphertext block, C_i, is successively fed back to the input of the algorithm, after having been delayed for 64 bits by the feedback register, R. The output of this register, C_{i-1}, is added (modulo-2) with the next 64-bit input plaintext block, P_i. Refer to Figure 2.1, below, for an illustration of the DEA in CBC mode. This results in an inter-block dependency, thus precluding undetected additions or removals of ciphertext blocks. The cryptovariables here are the initial value of the feedback register, I_0, and the cryptographic key used for the DEA, K_i. Other advantages are that the last 64-bit ciphertext block serves as a checksum to test the integrity of the whole message, and that the initial value of the feedback block to be added (modulo-2) with the first plaintext block may be selected to be either a random number (to provide extra random perturbation to the whole ciphertext message) or a time stamp / serial number (to prevent any replay attack).

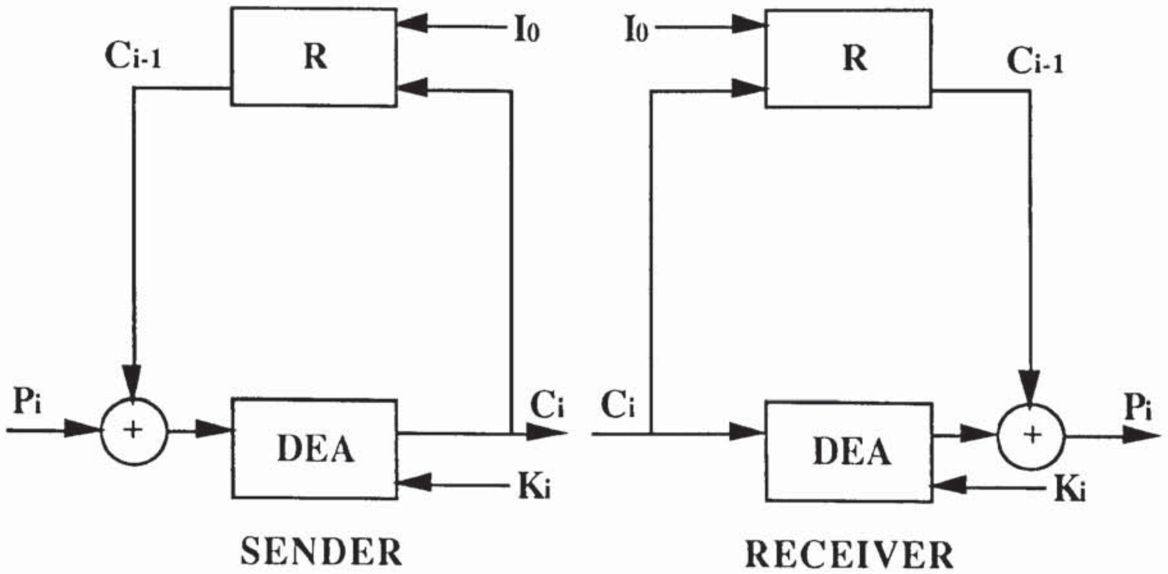


Figure 2.1 Operation of the DEA in CBC mode

2.4.3.3 Output Feedback Mode (OFM). This mode of DEA operation produces a synchronous stream cipher, by means of passing the contents of a 64-bit input register, R, into the DEA as plaintext, the corresponding 64-bit output ciphertext block being fed back to the input register for successive iterations of encryption. The least significant bit of the output block of each iteration is used to encrypt the plaintext bit stream, P_i, in a bit-wise (modulo-2) addition, to produce a ciphertext bit stream, C_i. Refer to Figure 2.2, below, for an illustration of the DEA in OFM mode. The cryptovariables here are the initial value of the input register, I_0, and the cryptographic key used for the DEA, K_i. This mode of operation has the advantages of no error propagation and the ability to "sign" messages with a time-stamp/serial number or with a random perturbation. The disadvantage of this mode of operation is that data integrity cannot be guaranteed, since data blocks may be modified without knowledge of the cryptovariables. Note that in using this mode of operation, however, if ciphertext blocks are added or deleted then this will be apparent, since cryptosynchronization will be lost at the receiver.

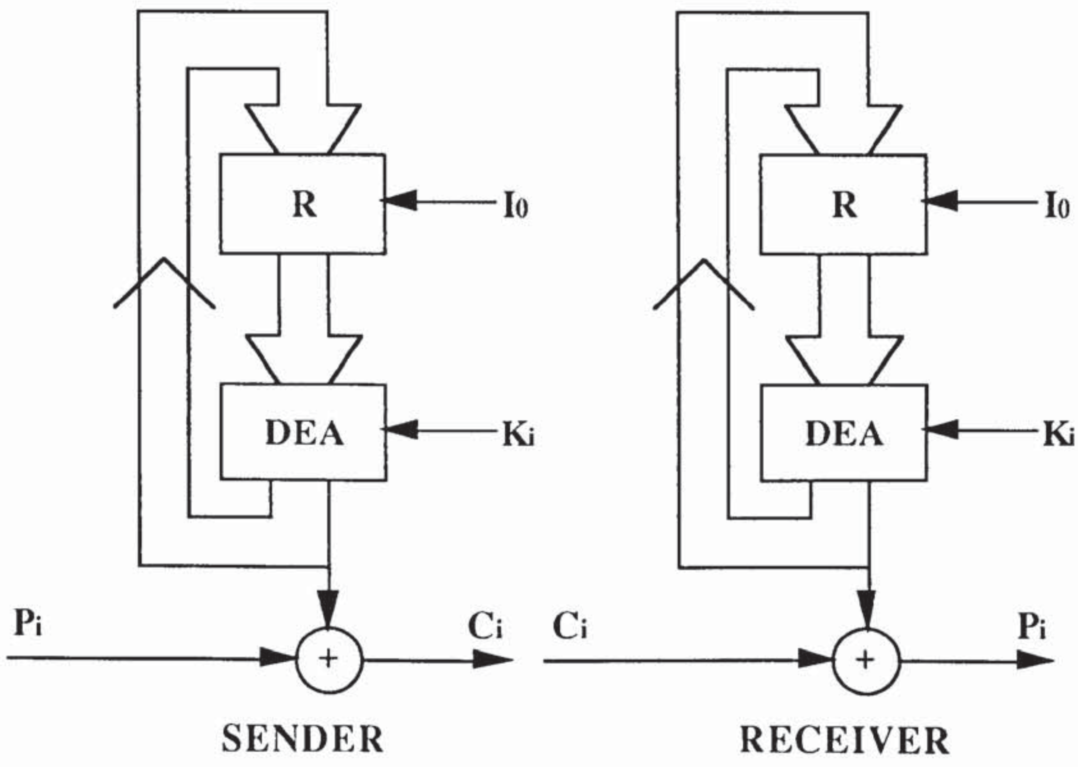


Figure 2.2 Operation of the DEA in OFM mode

2.4.3.4 Cipher Feedback Mode (CFM). This mode of DEA operation produces a synchronous stream cipher, by means of passing the contents of a 64-bit shift register, R, into the DEA as plaintext, all resultant ciphertext being discarded except the LS bit which is added (modulo-2) bit-for-bit with the plaintext bit stream, P_i. This constitutes the ciphertext bit stream, C_i, which is also shifted into the MS bit of the input register, R, upon each iteration of the DEA (that is, for each bit of ciphertext). Refer to Figure 2.3, below, for an illustration of the DEA in CFM mode. The cryptovariables here are the initial value of the input shift register, I_O, and the cryptographic key used for the DEA, K_i. This mode of operation has the advantages of no error propagation beyond a given DEA block, and the ability to "sign" messages with a time-stamp/serial number or with a random perturbation. The disadvantage of this mode of operation is that data integrity cannot be guaranteed, since data blocks may be modified without knowledge of the cryptovariables. Note that in using this mode of operation, however, if ciphertext blocks are added or deleted then this will be apparent, since cryptosynchronization will be lost at the receiver.

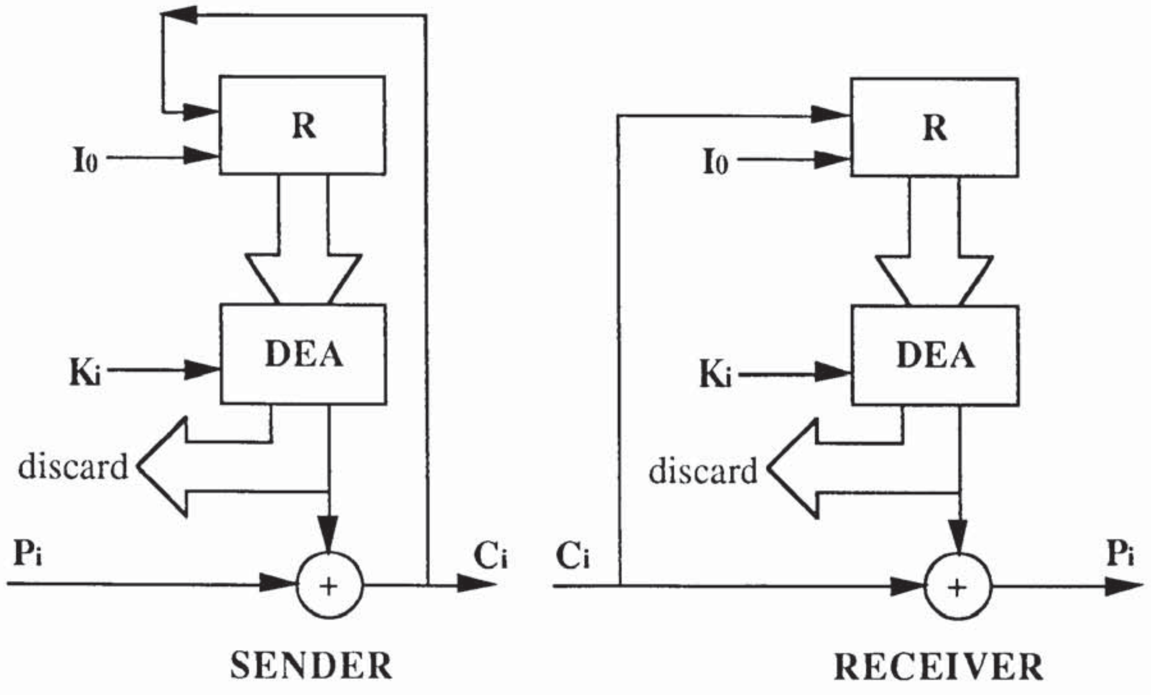


Figure 2.3 Operation of the DEA in CFM mode

2.4.4 Practical Implementation of the DEA.

The implementation of the DEA is a relatively straightforward matter, given the iterative nature of the algorithm. The algorithm may be implemented on a pipeline processor resulting in a high speed solution. Some typical examples of DEA integrated circuit solutions are shown in Table 2.1, below.

Note that the DEA may also be implemented in software for a more general purpose but slower solution. An example of this is a 'C' version of the DEA written by Brown [22]. The speed of operation of this will depend upon the nature of the processor on which the machine code is executed. Refer to Section 1.6 for details of the security matters to be considered in implementing such a cryptosystem.

Manufacturer	Device Number	Speed (ECB mode)
Intel™	8294A	3.2 Kb/s
Texas Instruments™	99541	3.2 Kb/s
Motorola™	6859	400 Kb/s
Western Digital™	2001 / 2002	1.3 Mb/s
Fairchild™	9414	13.3 Mb/s
AMD™	9518 / Z8068	14 Mb/s

Table 2.1 DEA Encryption Manufacturers, Devices and Speeds

2.5 The Current Status of Symmetric Algorithm Cryptosystems.

There follows a summary of the development of symmetric algorithm cryptosystems pertinent to applications in mobile communication systems.

2.5.1 Current development of private key encryption schemes.

Private key cryptography has seen a marked decline in research and development since the 1980s, in favour of public key (asymmetric algorithm) cryptography and zero-knowledge schemes.

A major example of development since the DEA is the FEAL. The development of FEAL resulted from the NSA withdrawing the relevant approval of the DEA in 1986.

FEAL (The Fast Data Encipherment Algorithm) was proposed by Shimizu and Miyaguchi [23] in 1987. It is suitable for both hardware and software implementation, even on personal computers. The algorithm has similarities to the DEA, but the S-boxes of the DEA are replaced by S-functions, defined as :-

$$s(x, y, \delta) = \text{ROL } 2((x + y + \delta) \text{ MOD } 256) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.23$$

where :- x, y : one byte of data
δ : constant (0 or 1)
ROL 2 : 2-bit circular left shift

The security of this S-function is required to afford at least the level of security afforded by the S-box of the DEA.

The FEAL is ideal for implementation on pipeline processors, since it processes 64-bit blocks of plaintext via four successive iterations of a product cipher using 16-bit intermediate keys derived from a 64-bit key via six iterations of a key-generation function. The four iterations of the product cipher are both preceded and succeeded by a stage of gross perturbation by addition (modulo-2) with a selected concatenation of the 16-bit intermediate keys.

Another major example of symmetric algorithm development for the purposes of encryption (ciphering) is the A5 class of algorithms used in the ETSI GSM900/DCS1800 [24] and DECT [25] personal communication systems. Refer to Chapter Five for further discussion of these algorithms.

2.5.2 Current development of private-key authentication schemes.

The properties of private-key (symmetric algorithm) cryptosystems and the requirements of authentication are largely incompatible. The fundamental requirement of authentication is that the party to be authenticated is able to demonstrate possession of knowledge that only that party can possess. The problems that occur with the use of symmetric algorithms for the purposes of authentication are, firstly, that shared secret cryptographic keys allow secret authentication data to be determined and, secondly, that it is not feasible to securely distribute secret keys between a large number of communicating parties for the purposes of catering for all possible communication sessions between those parties.

Given the above incompatibility, however, private-key cryptographic algorithms still find use in authentication schemes. Considering the (relative) ease with which symmetric algorithms may be used to encrypt data, then if their cryptovariables can be known by a trusted host they can be used to secure secret user authentication data whilst it is transmitted over an insecure channel. A typical example of this is in electronic funds transfer, where many financial institutions use the DEA to protect communications between their host computers and remote devices such as automatic teller machines, both of which have a common shared key. Such communications will cater both for user authentication and for facility attribution.

Note that any symmetric algorithm that has its cryptovariable chosen by a trusted host may be used by that host for the purposes of authentication and for the provision of data integrity. Such a use of symmetric algorithms is possible since, to an attacker, such an algorithm will appear to be a one-way function even if, for the host, it is reversible.

2.5.3 Current development of private key data integrity schemes.

Data integrity can be provided in one manner or another by any cryptographic algorithm capable of encrypting; if the data to be protected is passed through the particular algorithm, the resultant data will be deterministic upon the input data and will only be able to be generated by encryption using the required cryptovariable. In this manner data integrity schemes are synonymous with digital signature schemes. A problem occurs in the use of this simplistic approach to assessing data integrity, however, in that the check data generated would be at least as large as the data being protected. Two ways in which this problem may be solved are by the use of cipher feedback mechanisms or by the use of hashing functions.

An example of the use of cipher feedback for the purposes of data integrity is as explained in Section 2.4.3.2; the CBC mode of operation of the DEA. Note that the use of cipher block chaining results in an inter-block dependency, thus precluding undetected additions or removals of ciphertext blocks. The last 64-bit ciphertext block serves as a checksum to test the integrity of the whole message, irrespective of the message length. Note that in this example, data integrity protection occurs as a consequence of encryption for the provision of message secrecy. Note also that any encryption algorithm may be configured to incorporate cipher block chaining and, as such, may be constitute a cryptographic data integrity scheme. It is for this reason that cryptographic algorithms do not need to be designed solely for the protection of data integrity.

Hashing functions are one-way functions which take all or part of an input data field, add any data padding as necessary to achieve a required size for the field to be hashed and then perform hashing. Hashing is achieved by selecting members of a data field and cross-mapping them into the members of another data field, which may be either the same size or smaller, typically the latter. This operation is typically repeated in an iterative manner, and may be dependent upon cryptovariables such as a cryptographic key and an initial perturbation value. In this manner, the mappings are deterministic but appear to be random in nature. Any general hash function, #(M), must have the property that if a set of messages, M, is known together with their #(M) values it is nevertheless extremely difficult, given some other value, $\#_o$, to find any message, M, such that $\#(M) = \#_o$. Thus it is required that # must be large enough to prevent the compilation of a table of M and #(M) to attempt to find a value of M for any given #(M). Such one-way hash functions have several uses, but may not be suitable for authentication.

An example of a simple hash function is shown in Davies and Price [26]. The essential property of #(M) is that it is one-way. In this example the function # is public, hence an attacker may calculate #(M) for any given message, M. The security afforded will be broken if the attacker can find another message M' such that #(M') = #(M). Given that the range of #(M) is much smaller than that of M, then there must be a large number of solutions, but no *systematic* way of finding one. The authors suggest the use of a concatenated hashing function of the form :-

$$y = Ek(x) \oplus x \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 2.24$$

where	E	: encryption function
	k	: cryptovariable
	x	: input to function
	y	: output from function
and	$\oplus$	: bit-wise modulo-2 addition

Note that given y, this equation cannot be solved for k and x jointly. Since k is a key, y is thus a one-way function of it. It is a one-way function of x for all k, since x enters twice in the equation. This is on the basis that Ek(x) is a strong cipher.

The above equation can be used as a module for one-way message encryption, by taking a message M and splitting it into n blocks of (say) 56 bits, thus $M = M_i$, for $i = 1$ to $n-1$. These message blocks are used as cryptovariable input k_i to successive one-way modules, the output y_i of each module being the input x_{i+1} to the next module. Thus, given some initializing value for x_1, (which may be message derived), y_i is a one-way function of the message M. This is because, given y, the equation cannot be solved simultaneously for k and x. The output y must be a one-way function both of k, (since k is a key) and of x, since x enters twice in the equation. This is based on the assumption that Ek (x) is a strong cipher, such as the DEA in CBC mode.

A useful set of criteria for use in the practical implementation of a cryptographic algorithm is that of the Cryptographic Finite State Machine (CFSM), as demonstrated by Pichler [27]. The CFSM model caters for the generation of stream ciphers, block ciphers and hashing functions. The finite state machine is loaded with a specified initial value (the cryptovariable) into its state register and is then iterated whilst its key register is loaded with data generated using bits selected from the message and padding bits. After a given number of iterations, all or part of the state register is taken as the hashed result.

Note that a hash function that is to be used for cryptographic purposes must be collision free; this means that to find two different messages M_1 and M_2 that have the same hash result, requires a computational effort of the order of $2^{n/2}$ applications of the hash function.

The bit selection and key loading mechanisms are essential to the operation of the hash function. It is important that each message bit should appear in the key register several times during the hashing operation, in order to ensure that the contents of the key register cannot be chosen during a specific iteration of the state machine, without affecting it during other iterations. State invertibility of the CFSM function is desirable, since it it will ensure that there are no intermediate hash results with the same successor state for a given key. Key injectivity is desirable since it ensures that intermediate hash results are dependent upon the selected message bits.

2.5.4 Current development of private key distribution schemes.

An interesting development in this area has been the idea of using private-key certificates, as proposed by Needham and Schroeder [28]. Consider a cryptosystem for communications between parties A and B, via an intermediary I. The parties A and B are required, when using conventional cryptography, to share a secret key, but this can be avoided by using a trusted intermediary who is in possession of the secret keys of both parties. This allows party B to encrypt a message m for the intermediary only, who may then translate the message into an encrypted version which can be recovered by party A, thus :-

$$B \rightarrow A : \{A, m\}k_{b,i} \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 2.25$$

$$A \rightarrow I : \{A, m\}k_{b,i}, B \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 2.26$$

$$I \rightarrow A : \{m, B\}k_{a,i} \quad \dots\dots\dots\dots \quad 2.27$$

The key $k_{b,i}$ is known only to B and I. I receives receives and decrypts the message m addressed to A, then re-encrypts it with the key of A, $k_{a,i}$. The address of the message will be changed from A to B. Note that parties A and B can use a time stamp to prevent replay of their messages.

Since it is not feasible for the intermediary I to securely maintain a database large enough to hold the keys of all the parties, the database is dispersed using the protection of a master key. Now party A must provide both its key and the key of party B to the intermediary I, thus :-

$$B \rightarrow A : \{A, m\}k_{b,i} \quad \dots\dots\dots\dots \quad 2.28$$

$$A \rightarrow I : \{A, m\}k_{b,i}, \{k_{b,i}, B, L_b\}k_i, \{k_{a,i}, A, L_a\}k_i \quad \dots\dots \quad 2.29$$

$$I \rightarrow A : \{m, B\}k_{a,i} \quad \dots\dots\dots\dots \quad 2.30$$

The key k_i is the master key of I, known only to I and its clones. The encrypted key $\{k_{b,i}, B, L_b\}k_i$ is the private-key certificate of party B. This is a message published from I to B to inform B that $k_{b,i}$ is the key of B during the lifetime of the certificate, L_b. Note that party A need not request translation; party B or any third party with access to the certificate directory can make the request. The intermediary would, however, refuse to perform the translation if a message is not addressed to the owner of a target certificate. Note that chosen-plaintext attacks on key $k_{b,i}$, for example, may be thwarted by use of a single-use key, $\dot{k}_{b,i}$, to encrypt a message. This is achieved by B selecting the lifetime of the key, $\dot{L}_b$, to span between encryption of the message from B and the translation request of A.

Key distribution is supported by private-key certificates, similar in nature to the Kerberos authentication system as proposed by Miller *et al* [29], thus :-

$$B \rightarrow I : \{k_{b,i}, B, L_b\}k_i \quad \dots\dots\dots\dots \quad 2.31$$

$$I \rightarrow B : \{\dot{k}_{b,i}, B, \dot{L}_b\}k_i, \{\dot{k}_{b,i}, I, \dot{L}_b, \text{checksum}\}k_{b,i} \quad \dots\dots \quad 2.32$$

Thus B receives a new certificate containing a fresh key and a separate copy of the same key with a checksum of the new certificate, encrypted in the old key of B, $k_{b,i}$. This allows party B to calculate the same checksum and, comparing with that received, to prove it was party I who encrypted the certificate.

There are variants of this protocol to allow an administrator to assign an initial key and a certificate to a new user, and to allow parties A and B to directly share a key. Note that the incorporation of a new user will require initial secure communication between the administrator and the new party. The protocol can also be extended to allow different intermediaries to share secret information, either via a higher intermediary or via public-key cryptography.

Any encryption algorithm may be used for the distribution of session keys. If the distribution algorithm is symmetrical, however, then there will have to be an agreed master key, common between the parties which are to share session keys. This is an unavoidable consequence of using symmetric algorithms to distribute secret information between parties which must, therefore, have previously agreed some other secret key.

2.6 Conclusions.

This chapter has outlined an investigation into the theory pertinent to secrecy systems. Shannon's theory of secrecy systems has been explained in relation to the entropy of cryptovariables, rate of language, redundancy of language, equivocation of a cryptosystem and unicity distance. The distinction between perfect secrecy and cryptosecrecy has been explained and it has also been shown that any real cryptosystem must rely on cryptosecrecy. It has been shown that cryptosecrecy can be expressed in terms of work factor and cover time.

The general authentication security requirements have been considered with regard to user authentication categories, user authentication requirements and system authentication requirements. The various authentication security requirements have also been considered with regard to the protection they provide to the user and to the general category of security mechanisms which may need to be implemented in order to provide for such authentication security requirements. It has been shown that several categories of authentication security must be provided for, in order to afford comprehensive protection to the system against attack.

Symmetric algorithms and private-key cryptography have been considered with regard to the properties of symmetric algorithms, namely their character distribution and cryptanalysis and their error propagation properties. The uses of symmetric algorithms in private-key cryptosystems have been discussed, with particular reference to pseudo-random key streams and to block ciphers. The modes of operation of block ciphers have been considered in detail, with a particular example being made of the DEA. The effects of the DEA on plaintext data have been considered, as has the practical implementation of the DEA. The limitations of the DEA have been mentioned as an example of the limitation of the use of any (block) symmetric algorithm cryptosystem. It has been shown how the mode of operation of a block cipher must be selected to suit the environment in which it is to operate. It has also been shown how symmetric algorithms may be used to mimic the properties of one-way functions and may be used in key distribution protocols. The latter application has been stated with regard to the concept of private-key certificates.

It is evident that the development of symmetric algorithms has declined over the last decade, with a greater emphasis being put on asymmetric algorithm cryptography, such as public key systems and zero-knowledge schemes. Refer to Chapter Three for a detailed consideration of asymmetric algorithm cryptography. The reasons for such a decline in the development of symmetric algorithm cryptography are threefold. Firstly, symmetric algorithm cryptography is conventional and employs techniques which are well understood and thoroughly investigated, such as alphabetic substitution and transposition. This leaves less scope for new developments. Secondly, major bodies such as the US NSA have introduced certified cryptosystems for approved applications. The DEA is the most common example of this. Its certification, however, has now been withdrawn in favour of new algorithms. Thirdly, the significant practical developments in symmetric algorithm cryptography within the last decade have largely been associated with the advent of the Second-Generation of PLMN systems. This development has been a highly specialist application, and has not been available for public scrutiny.

In conclusion, it can be seen that there are two fundamental properties which the cryptosystem designer must take into consideration when deciding whether to choose a symmetric algorithm in favour of an asymmetric algorithm, and when deciding the nature of any symmetric algorithm. Firstly, the designer must assess what effects there will be on the nature of the communication system due to the cryptographic protocols inherent in a cryptosystem which requires there to be an initial shared secret, common to communicating parties. Secondly, the designer must assess an appropriate compromise between the level of cryptosecrecy required and the processing and signalling overheads resulting from the use of the particular algorithm in providing the chosen level of cryptosecrecy.

CHAPTER 3. ASYMMETRIC ALGORITHM CIPHERS AND CRYPTOSYSTEMS.

This chapter investigates the particular characteristics and uses of asymmetric algorithm (public key) cryptography. The various applications of asymmetric algorithm cryptography are examined, with particular emphasis on both digital signature schemes and cryptographic key distribution schemes.

The RSA scheme is used as a particular example and the associated requirements of prime number generation are presented. The Fiat-Shamir zero-knowledge authentication scheme is examined in detail and is compared to the RSA scheme. Practical implementations and their associated security levels are investigated.

The implications of the requirements of both cryptographic key management and cryptosynchronization are investigated, as are the significance of both signalling and processing overheads.

The current status of asymmetric algorithm cryptography is examined with consideration being given to encryption schemes, to authentication schemes, to data integrity schemes and to cryptographic key distribution schemes.

3.1 Asymmetric Algorithms and Public Key Cryptography

As explained in Chapter Two, asymmetric algorithms are conventionally used to encrypt plaintext into ciphertext for secure storage and transmission using a public encryption key, and to decrypt the ciphertext to recover the original plaintext using a secret decryption key. Due to the fact that the cryptographic keys used for encryption and decryption are dissimilar, the encrypting key used is commonly known both by the encrypting party and by the decrypting party, but the decrypting key is known just by the decrypting party. Thus the decrypting key is secret (private). Such public key cryptographic algorithms may also be used to implement digital signature schemes, refer to Sections 3.1.2.1 and 3.2.3.

3.1.1 Properties of Asymmetric Algorithms.

Asymmetric cryptographic algorithms, when used to encrypt plaintext data, must have two essential properties; they must not allow the recovery of plaintext from ciphertext using the same cryptographic key that was used to originally generate the ciphertext from the plaintext, and the ciphertext generated must be of such a nature that it is an intractable problem either to recover the plaintext from the ciphertext without the appropriate cryptographic key, or to recover that cryptographic key from the plaintext. Note that the latter property of intractability is due to the randomizing effect of a cipher on plaintext and is an essential property of any type of cryptographic algorithm and its key.

There are other uses for asymmetric cryptographic algorithms, and other desired properties associated with these uses. Details of such other uses are given in Sections 3.1.2.1, 3.1.2.3 and 3.2.2.

3.1.1.1 Distribution and cryptanalysis. As with symmetric algorithms, asymmetric algorithms act on input plaintext data so as to modify the distribution of character frequencies within the ciphertext data, refer to Section 2.2.1.1. The essential difference of asymmetric algorithm cryptography is that the key used to encrypt data must not be able to be used to recover the plaintext data, moreover, there may not be any key at all which can be used to recover the plaintext data, as is the case for one-way functions; refer to Sections 2.5.3 and 3.1.2.3.

3.1.1.2 Error propagation. As is the case for symmetric cryptographic algorithms, error propagation is an important consideration in asymmetric algorithm cryptosystems, refer to Section 2.2.1.2. The effects of error propagation in block ciphers is of particular relevance in public key cryptography since such algorithms tend, by their intrinsic nature, to operate in a block fashion. This is due to the typically large size of data items processed by such algorithms. For example, the RSA algorithm may operate on data blocks of a typical size of 512 bits; refer to Section 3.2.

3.1.2 Uses of Asymmetric Algorithms in Public Key Cryptosystems.

There are several uses for asymmetric cryptographic algorithms in cryptosystems. The common use of public key encryption has been outlined in Chapter One. There are other uses of such algorithms, including digital signature schemes and one-way functions. These are broadly considered below.

3.1.2.1 Digital signatures using public key ciphers. The essential difference between symmetric algorithm schemes and asymmetric algorithm schemes is that the latter allow the implementation of a public key / private key schemes. These have the intrinsic advantage of being able to be used not just for message encryption schemes, but also for authentication schemes.

The basis for a digital signature scheme is that it shall be possible for easy public verification of a signature, but it shall be unfeasible for any party other than the signatory to generate the signature. This is commonly achieved by means of a cryptographic challenge-response exchange, where a signatory (the authenticator) is challenged to sign (random) data supplied by the verifier, and return it to the verifier. The signature is generated using secret data known only to the signatory, who publishes data to allow the public verification of any signature in such a manner that the determination of the secret data, from any public data or from any signature, is an intractable problem.

3.1.2.2 Digital signatures using private key ciphers. This is considered for the purposes of comparison with the use of public key ciphers. Signature schemes can also be implemented using private key (symmetric algorithm) ciphers. These may be implemented where the typically reduced processing requirement of such symmetric algorithms in comparison to asymmetric algorithm ciphers is desired, but the added advantage of using asymmetric algorithms to generate a cryptosystem from a no-shared-secrets state is not required. Note that the processing overheads of asymmetric cryptographic algorithms (such as RSA) may require an increase in processing of the order of 1,000 times that of a symmetric cryptographic algorithm (such as the DEA). To implement a symmetric algorithm signature scheme, the data to be signed is passed through a symmetric algorithm using a secret encryption key. Some, or all, of the resultant ciphertext is then used as the signature. This application of symmetric algorithms is popular in secure data integrity schemes, where it is commonly referred to as a cryptographic check-sum.

The intrinsic requirements of a secure data integrity scheme are threefold; namely that it ensures that each bit of data to be protected affects any cryptographic check-sum in a maximal manner, that the check-sum cannot be generated without possession of a secret key, and that it is not possible to determine the secret key given the protected data and the integrity-related data.

3.1.2.3 The uses of one-way functions in authentication. One-way functions have a particular use in authentication; both in data authentication (integrity and non-repudiation) and in party authentication (proof of identity).

The essential property of the one-way function is that it has an output which is deterministic with respect to the data input to the function, but that the output data cannot be used to derive the input data. This can be achieved by mapping the input data into a space which is significantly smaller than that of the input data, it can also be achieved by the use of a cryptographic algorithm (be it symmetric or asymmetric) which utilizes a particular cryptovariable. Such a cryptovariable may be public if the algorithm is asymmetric, or secret if the algorithm is symmetric.

A typical use of a one-way function is in party authentication, where it may be used to implement a digital signature scheme. A popular mode of such a scheme is the cryptographic challenge-response; refer to Chapter Five.

3.1.3 Prime Numbers and their Generation.

A crucial aspect in the implementation of many public-key cryptosystems is the generation of prime numbers, since these are fundamental to the operation of many such cryptosystems, due to an intrinsic difficulty in factorizing a composite number which is a product of two large primes, refer to Garey and Johnson [30].

The problems associated with the generation of large primes (that is, in excess of 100 digits) has been investigated in great detail. A probabilistic method of searching for prime numbers has been developed, refer to Chapter Six. The problems associated with finding large primes are that the numerical processes are computationally intensive and their specific magnitude is unpredictable: a Sun™ *Sparc 10* workstation took somewhere between 8 minutes and 23 minutes to generate a 40-byte prime.

3.2 The RSA Public Key Cipher.

The Rivest-Shamir-Adleman public-key cryptosystem [31] is universally accepted as the bench-mark in public-key cryptography; comparable to the status of the DEA in private-key cryptography. First published in 1978, it broke new ground in asymmetric algorithm cryptography. It is considered here to illustrate the properties of such cryptosystems and to allow comparison with other developments in the field of asymmetric algorithm cryptography.

3.2.1 Mathematical basis for the RSA algorithm.

The security of the cipher is based upon the difficulty of two numerical problems; the discrete logarithm problem and the factorization problem, refer to Odlyzko [32]. In this cryptosystem, the messages, cryptograms and cryptovariables all belong to a set of integers:-

$$Z_N = \{1,\dots, N-1\} \qquad \dots\dots \quad 3.1$$

where $N = p \times q$

and p,q are primes

The set Z_N and the product N are publicly known. The enciphering transformation is defined as:-

$$C = E_{K_e}(M) = M^{K_e} \pmod N \qquad \dots\dots \quad 3.2$$

where M = plaintext message

and C = resultant cryptogram

E = enciphering transformation

K_e = public enciphering key

The corresponding deciphering transformation is defined in equation 3.3, below :-

$$M = D_{K_d}(C) = C^{K_d} \pmod N \qquad \ldots\ldots\ldots\ldots \quad 3.3$$

where D = deciphering transformation
and K_d = secret deciphering key

The plaintext message transformed into a corresponding cryptogram must be able to be re-transformed in order to recover the original plaintext message, thus:-

$$M = D_{K_d}(E_{K_e}(M)) \qquad \ldots\ldots\ldots\ldots \quad 3.4$$

Thus substituting for C in equation 3.3 from equation 3.2 we get :-

$$(M^{K_e})^{K_d} = M \pmod N \qquad \ldots\ldots\ldots\ldots \quad 3.5$$

It can be shown that if the integer N is prime, the congruence 3.5 will have a solution if and only if :-

$$K_e \times K_d \equiv 1 \pmod{N-1} \qquad \ldots\ldots\ldots\ldots \quad 3.6$$

but $N = p \times q$ 3.7

where p and q are primes,

thus the congruence 3.5 has a solution if and only if :-

$$K_e \times K_d \equiv 1 \ (\bmod\ \gamma(N)) \qquad \ldots\ldots\ldots\ldots \quad 3.8$$

where $\gamma(N) = \mathrm{lcm}(p-1, q-1)$ 3.9

and lcm is the lowest common multiple.

Thus the receiving party creates the system by :-

- Selecting K_e at random and publishing it.
- Selecting the secret pair of primes {p, q} and publishing their product, N.
- Calculating the secret key, K_d, by solving the congruence 3.8.

Proof of the congruence 3.8 is given in Seberry and Pieprzyk [33].

3.2.2 Cryptographic key distribution using the RSA cipher.

Using the RSA cipher as a public key cryptosystem, it is possible to distribute (secret) cryptographic keys in a secure manner. This is achieved by establishing a receiver-generated public key cryptosystem as defined in Section 3.2.1; the sending party / parties then distribute their (secret) cryptographic keys by encrypting them using the particular public key and transmit the resultant cryptogram. This cryptogram can then be decrypted by the receiving party using the secret decryption key, thus recovering the original secret ciphering key sent by the transmitting party. In this manner, (secret) cryptographic keys may be securely distributed between communicating parties over an insecure channel.

3.2.3 Digital signature scheme based on the RSA cipher.

The implementation of a digital signature scheme using the RSA public key cryptosystem is achieved by means of the same receiver-generated cryptosystem as defined in Section 3.2.1, but by the receiving party publishing the decryption key, K_d, and keeping secret the corresponding encryption key, K_e. In this manner, a party to be authenticated (the receiving party) may encrypt data using its secret encryption key and any number of parties may verify the authenticity of that party by using the public decryption key. Thus a typical authentication exchange may be as follows :-

- The party to be authenticated generates a public key cryptosystem as defined in Section 3.2.1, and publishes the decryption key K_d and the modulus N.
- A verifying party sends a random vector M to the authenticating party, which signs it using the secret encryption key K_e (as defined in equation 3.2) to produce a signature vector C. This signature vector is then returned to the verifying party.
- The verifying party then decrypts the received signature vector C (as defined in equation 3.3) to produce the (random) vector M'. This vector M' is compared to the random vector M sent to the authenticating party. If the two vectors are identical, then the authenticity of the identity of the authenticating party is considered to have been positively verified.

3.2.4 Practical considerations for implementing an RSA cryptosystem.

As with the implementation of any cryptosystem, there are several factors to be considered including the security level provided by the particular cryptographic algorithm and cryptovariable used, and the amount of processing required to implement this. These are considered in Chapter Six. There is, however, a particular consideration required regarding the security inherent within the nature of the RSA algorithm; that is the manner in which messages are concealed by the cryptosystem.

The RSA algorithm has a characteristic feature in that it does not always hide a message. An example of this is given below :-

Consider an RSA cryptosystem where the sending party has the public key $K_e = 17$ and the modulus $N = 35$. If a message includes numbers from the set S_X, then the cryptograms will be equal to the plain messages, where :-

$$S_X = \{ 1, 6, 7, 8, 13, 14, 15, 20, 21, 22, 27, 28, 29, 34 \} \quad \ldots\ldots\ 3.10$$

Thus $1^{17} = 1 \pmod{35}$, $6^{17} = 6 \pmod{35}$, ..., and $34^{17} = 34 \pmod{35}$

A more severe example is shown where $K_e = 865$ and $N = 10{,}573$. Here it can be shown that there is no concealment of any messages, since :-

$$M^{865} = M \pmod{10{,}573} \qquad \text{for all } M$$
$$\text{where } K_e = 865 \text{ and } N = 10{,}573$$

This property is explored by Blakley and Bolosh [34], and is shown to be due to the fact that any arithmetic modulo-N, where :-

$$N = p \times q \quad \ldots\ldots\ 3.11$$

where p and q are primes

will always have the property that at least nine messages, M, remain unaffected by being raised to a positive odd integer K_e. For these particular values of M :-

$$M^{K_e} = M \pmod{N} \quad \ldots\ldots\ 3.12$$

3.2.5 Practical Implementation of the RSA Algorithm.

There have been several practical implementations of the RSA algorithm; refer to Section 3.5.1.1.

3.3 The Fiat-Shamir Scheme for Authentication and Digital Signature.

Zero-knowledge cryptosystems allow users to communicate with a given party in a secure manner without necessarily being privy to any information which could prove to be of use to any potential attacker of a secure communications system. A typical example of this is the Fiat-Shamir cryptosystem [35], which is detailed below.

3.3.1 Mathematical basis for the Fiat-Shamir algorithm.

The Fiat-Shamir zero-knowledge signature scheme provides authentication by means of the problem of factoring. Consider a cryptosystem where :-

A is a Prover, **B** is a Verifier, **M** is a Message and **I** is Identification Data. Let there be a vector of **k** secret values $\mathbf{V_j}$ formed as :-

$$V_j = f(I, C_j) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.13$$

for $\mathbf{j} = 1,2....\mathbf{k}$

where $\mathbf{C_j}$ is a set of small chosen integers to make each $\mathbf{V_j}$ into a quadratic residue

and f is a one-way function

A stores secret values $\mathbf{S_j}$ such that :-

$$S_j^2 = v_j^{-1} \bmod (m) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.14$$

The signature process of **A** is defined as :-

(1) Find **t** random values $\mathbf{r_i}$ and calculate $\mathbf{x_i}$ where :-

$$x_i = r_i^2 \bmod (m) \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.15$$

(2) Compute a one-way function $f(\mathbf{M}, \mathbf{x_1}, \mathbf{x_2}, ...\mathbf{x_t})$ and use the first $\mathbf{k} \times \mathbf{t}$ bits for the matrix:-

$$| e_{ij} | = \begin{vmatrix} e_{11} \ldots\ldots e_{1k} \\ \ldots\ldots \\ e_{t1} \ldots\ldots e_{tk} \end{vmatrix} \quad \ldots\ldots\ldots\ldots \quad 3.16$$

(3) Calculate the signature (a vector of **t** values) :-

$$y_i = r_i \prod_{j=1}^{j=k} S_j^{e_{ij}} \bmod (m) \quad \ldots\ldots\ldots\ldots \quad 3.17$$

for $\mathbf{i} = 1,2,....\mathbf{t}$

The product is taken over all **k** values of $\mathbf{s_j}$, but only those selected where $\mathbf{e_{ij}} = 1$.

Note that :-

B receives **M**, **I**, $\mathbf{e_{ij}}$ and $\mathbf{y_i}$
B has stored **m** and $\mathbf{c_j}$

The verification process of **B** is defined as :-

(1) Calculate the vector :-

$$\mathbf{v_j} = f(\mathbf{I}, \mathbf{c_j}) \quad \text{.........} \quad 3.18$$

(for $\mathbf{j} = 1,2,....\mathbf{k}$)

(2) Calculate the vector :-

$$\mathbf{Z_i} = \mathbf{y_i^2} \prod_{\mathbf{j=1}}^{\mathbf{j=k}} \mathbf{V_j^{eij}} \bmod (\mathbf{m}) \quad \text{.........} \quad 3.19$$

(for $\mathbf{i} = 1,2,....\mathbf{t}$)

(3) Evaluate :-

$$f(\mathbf{M}, \mathbf{Z_1}, \mathbf{Z_2}, ...\mathbf{Z_t})$$

(4) Compare the first $\mathbf{k} \times \mathbf{t}$ bits with the given $\mathbf{e_{ij}}$ for identity.

3.3.2 A practical example of the Fiat-Shamir authentication scheme.

For example, let the field **I** be given by :-

I = (1 0 1 1 1 1 1 0 1 0 0 0 1 1 1 0 0)

where **I** represents user security-related parameters.

Let **m** = 35, and let there be a random function :-

$$\mathbf{V_j} = f(\mathbf{I}, \mathbf{C_j}) \quad \text{.........} \quad 3.20$$

The function *f* is described in table 3.1, which also includes suitable values for $\mathbf{v}_j^{-1}$ and $\mathbf{S}_j$.

j	v_j	quadratic residue (mod 35) ?	v_j^{-1}	S_j
1	2	no		
2	3	no		
3	5	no		
4	6	no		
5	2	no		
6	11	yes	16	4
7	4	yes	9	3
8	29	yes	29	8
9	18	no		
10	5	no		
11	11	yes	16	4
12	1	yes	1	1
13	10	no		
14	12	no		
15	8	no		

Table 3.1. Random function *f*, with suitable values for v_j^{-1} and S_j.

The resultant set is : (**I**, 4, 6, 3, 7, 8, 8, 4, 11, 1, 12)

This is equivalent to : (**I**, S_6, 6, S_7, 7, S_8, 7, S_{11}, 11, S_{12}, 12)

This set is then stored in an identity module such as a smart card. In this example, let $\mathbf{k}$ = 5.

In order to implement the Fiat-Shamir protocol, the following steps are undertaken :-

(1) **A** sends **I** to **B**.

(2) **B** generates $\mathbf{V_j} = f(\mathbf{I}, \mathbf{C_j})$ for **j** = 6, 7, 8, 11, 12 and renames these V_1, V_2, V_3, V_4, V_5.

(3) **A** picks a random positive integer $\mathbf{r}_1 \in [0, 35]$, for example $\mathbf{r}_1 = 6$, and sends $\mathbf{x}_1 = 6^2 \equiv 1 (\bmod 35)$ to **B**.

(4) **B** sends a random binary vector, for example (10111), to **A**.

(5) **A** sends to **B** :-

$$\begin{aligned} y_1 &= r_1 \prod S_j^{eij} \bmod (35) \\ &= 6 . S_1 . S_3 . S_4 . S_5 (\bmod\ 35) \\ &= 6 . 4 . 8 . 4 . 1 (\bmod 35) \\ &= 33 (\bmod 35) \end{aligned}$$

(6) **B** verifies that :-

$$\begin{aligned} x_1 &= y_1^2 \prod V_j^{eij} (\bmod 35) \\ &= 33^2 . 11 . 29 . 11 . 1 (\bmod 35) \\ &\equiv 1 (\bmod 35) \end{aligned}$$

Steps (3) to (6) are repeated for **i** = 1 to **t**. The value of **t** is selected such that **B** can be satisfied (probabilistically) that **A** actually possesses the knowledge it claims to have.

3.3.3 Practical considerations for implementing the Fiat-Shamir authentication scheme.

The validity of the attempted authentication of **A** is established by the verifier **B** if and only if all **t** checks are successful. A formal proof of the security of this scheme relies upon *f* being a truly random function and the modulus **m** being sufficiently large. This modulus will have to be factorized if the scheme is to be broken. If any particular practical implementation is vulnerable to an attack, then either **m** is too small or the function *f* is demonstrably non-random. Seberry and Pieprzyk [36] suggest that sufficient security will be provided by **m** being at least 512 bits long and f being provided by multiple use of the DEA. Considerations of the security level, and the signalling / processing overheads associated with this scheme are given in the following section.

3.4 Cryptosystem Issues

There are particular fundamental issues which have to be considered when implementing any cryptosystem, irrespective of whether it employs a symmetric algorithm or asymmetric algorithm cryptography. Six of these are identified below, for particular consideration with regard to symmetric and to asymmetric algorithms.

3.4.1 Security level.

This is the fundamental consideration in any cryptosystem, since the entire purpose of implementing such a system is to provide security. The security provided by symmetric algorithms such as the DEA is inherent in the randomizing effect the algorithm has on the plaintext data and in the work required to search the key space. Such key space exists by virtue of the number of possible key permutations, that is, the length of the particular cryptographic key used. For example, the security level afforded by the DEA is 2^{-56} for a 56-bit key. It is important to appreciate that this security level is entirely dependent upon the randomizing effect of the transpositions and substitutions inherent in the algorithm upon the plaintext; any successful cryptanalysis of the resulting ciphertext data will act to reduce the security level.

The security level of the RSA cryptosystem is determined by the size of the decryption key K_d on the basis that the modulus N, a product of two large primes, cannot easily be factored. If **n** is the binary size of K_d then the security level afforded is given by $2^{-\mathbf{n}}$. This is on the basis that the RSA algorithm cannot be otherwise weakened by some general form of attack. Two such generalized forms of attack are those proposed by Denning [37] and an iteration attack proposed by Simmons and Norris [38]. The first attack relies on determining a suitable instance for the solution of the discrete logarithm problem determined by the pair (K_e, C), where K_e is the encryption key and C is the ciphertext. The number of steps n required to perform the attack is given by :-

$$n \equiv O\left(e^{\sqrt{\ln N \ln(\ln N)}}\right) \qquad \dots\dots\dots\dots\dots\dots\dots\dots\dots\dots\dots \quad 3.21$$

For example, if N is 200 bits long, the discrete logarithm algorithm will require 2.7×10^{11} steps, whereas if N is 664 bits long, the same attack will require 1.2×10^{23} steps.

The second attack is an iteration based on knowledge of the triple (N, K_e, C). It has been shown by Rivest [39], however, that if the integers (p - 1) and (q - 1) contain large primes as factors, then the probability of a successful iteration attack $\simeq$ 0, if N is large. For example, if p and q are $\geq 10^{90}$, then for $0 \leq M \leq N-1$ (where M is the message), the probability of a successful iteration attack $\simeq 10^{-90}$.

The security level of the Fiat-Shamir zero-knowledge cryptosystem is provided by the difficulty in factoring **m**, a product of two large primes. The security level is given by $2^{-\mathbf{k}.\mathbf{t}}$ and is also dependent upon the function f being demonstrably random. For example, suppose the desired security level is 2^{-72}; this may be achieved by choosing **k** = 9 and **t** = 8.

3.4.2 Cryptographic key management.

The four fundamental aspects of key management, as identified in Chapter One, are random number generation, prime number generation, elimination of weak keys and key distribution. Not all these aspects of key management apply to all types of cryptosystem. Some examples are given below.

3.4.2.1 Random number generation. As with all cryptosystems, the cryptovariables should be random in nature, requiring the ability to generate likely random numbers and to test them for randomness. This may be done using purely digital means or by hybrid means, and may be performed in advance of such random numbers being actually required by the encryption algorithm. The DEA requires that a 56-bit secret key is generated for it to use as a cryptovariable for the purposes of each encryption or decryption. The RSA algorithm requires typical key lengths of at least 300 bits; the Fiat-Shamir algorithm also requires typical random numbers of length 300 bits.

3.4.2.2 Prime number generation. The generation of prime numbers is a complex process. A typical manner in which they may be generated is by means of a Jacobi function, as shown in Chapter Six. The DEA has the advantage of not requiring the generation of prime numbers. The RSA algorithm requires two primes per each selected modulus, although this modulus may be safely re-used for subsequent encryption or signature with a different random secret key. A typical size for such prime numbers in the RSA cryptosystem is at least 300 bits; the Fiat-Shamir cryptosystem also requires prime numbers of length at least 300 bits.

3.4.2.3 Elimination of weak keys. As with all encryption systems, the security provided is by virtue of the statistical redistribution of the ciphertext alphabet compared to the plaintext alphabet. As is shown in Section 2.4.4, in the RSA cryptosystem there exists particular combinations of modulus N and public key K_e for which there is no concealment of certain messages M. This requires that these particular combinations of N and K_e have to be inhibited. Similarly, the DEA suffers from weak keys, where there is a reduction in the mathematical complexity of the algorithm, and hence a reduction in its cryptographic strength. This occurs when the internally generated keys K(1) to K(16) are identical. This occurs when bits of *permuted choice 1* are all ones or all zeros. This gives rise to four parity-adjusted weak keys where there is, additionally, no difference between the operations of encryption and decryption. These are as shown below in Table 3.2 :-

01	01	01	01	01	01	01	01
1F	1F	1F	1F	0E	0E	0E	0E
E0	E0	E0	E0	F1	F1	F1	F1
FE	FE	FE	FE	FE	FE	FE	FE

Table 3.2 DEA Weak Keys (32-bit hexadecimal)

Further details of the associated semi-weak keys are given in [13].

3.4.2.4 Key distribution. This is a requirement for the secure dissemination of keys as necessary between the communicating parties of a particular cryptosystem. Cryptosystems can be divided into three categories regarding their relation to key distribution, namely :-

- Those which require secure distribution due to their keys being secret.
- Those which may be used to securely distribute secret keys.
- Those which cannot be used to distribute keys.

An example of the first category is the DEA. An example of the second category is the RSA cryptosystem. An example of the third category is the Fiat-Shamir signature scheme. Note that although symmetric algorithms such as the DEA may be used to distribute securely secret keys, they cannot do so without there already being in existence a common shared secret key. Asymmetric algorithm (public key) cryptosystems, such as the RSA cryptosystem, are able to distribute secret keys securely without any requirement for there being a prior common shared secret key.

A typical example of where the RSA cryptosystem may be used to securely distribute a common shared secret key between communicating parties is where a symmetric algorithm, such as the DEA, is being used to encrypt / decrypt communications between parties because of the relative processing efficiency of the latter. The RSA cryptosystem could be used initially to distribute the secret key for the DEA between the communicating parties without there being any initial common shared secret key. Such a key distribution algorithm, being asymmetric, is more process intensive than a symmetric algorithm, but would only be needed for the initial secure distribution of a secret key. Such an arrangement would constitute a hybrid cryptosystem.

An example of the application of a cryptosystem which cannot be used to distribute cryptographic keys is where authentication by signature only is required. Such a cryptosystem can be used to create the necessary signature vector from a given challenge, but cannot be used to distribute messages or keys due to the one-way (or hashing) nature of its data transformation. An example of such a cryptosystem is one which is used purely for access control.

3.4.3 Cryptosynchronization.

As explained in Chapter One, the encryption and decryption of data must be synchronized with respect to the ciphertext data; if data is lost from a ciphertext data file then this will result in the incorrect decryption of this data either from the point of occurrence of the ciphertext error and thereafter (for a stream cipher with cipher feedback), or for the data block concerned (for a block cipher). The maintainability of cryptosynchronization is not generally a problem for digital systems, since they are inherently synchronous and will typically be protected from any data loss by channel coding. An example of this is the GSM communication system, where a robust bit synchronization is provided by means of 64 frame synchronization bits in a 148-bit synchronization burst, specifically provided for this purpose. Additionally, there is a 26-bit training sequence within each 148-bit normal burst, which also provides intra-frame bit synchronization, refer to Chapter Five. Such a robust bit synchrony is provided because of the nature of the RPE-LTP speech codec, which requires that each of the (weighted) bits produced by such a speech coder in the transmitter be properly presented to the decoder in the receiver, refer to de Brito [40].

Cryptosynchronization in GSM is also aided by TDMA frame identity numbers, which are unique for each TDMA frame. Such a scheme uniquely identifies the 114-bit data fields (within each TDMA frame) produced by a coder, thus having two distinct advantages; the proper sequential order and continuation of the TDMA frame sequence produced by the coder is assured and TDMA frames may not be removed, added or re-played without the receiver becoming aware. Refer to Chapter Five for full details of the GSM security scheme.

As explained in Chapter Five, the GSM cryptosystem combines a stream cipher, used for its speed of operation and error non-propagation properties, with an overall block data format, used for robust channel coding to provide bit synchronization and error detection/correction. Since the TDMA frames are of fixed length, then any addition to or removal from the given block size will be prohibited. Moreover, should a TDMA frame be lost, for example through channel fading or through channel noise, then synchronous decryption will still be able to continue since the stream generation at both coder and decoder would remain unaffected and the TDMA frame identity would be used to continue decryption from the start of the next recovered TDMA frame. Such a self-synchronizing cipher is essential for the maintenance of speech quality over a noisy and fading mobile channel.

3.4.4 Error Propagation.

As explained in Chapter One, an important consideration in the design of any cryptosystem is the manner in which errors propagate within the cryptosystem and the effect such errors have on the operation of the cryptosystem. Any encryption algorithm which employs cipher feedback or, for that matter, any block cipher, will be prone to error propagation to a certain extent. Communication systems which use channels prone to errors in transmission are particularly susceptible to any effects of error propagation, since frequent losses of data over a poor channel may result in a frequently increased loss of recovered data due to error propagation. This is particularly the case in mobile radio communication systems.

Error propagation may be controlled in cryptosystems employing block ciphers by limiting the size of data blocks. Cryptosystems employing cipher feedback are less likely to be suitable for use over such channels. Robust channel coding schemes for the detection and correction of errors are essential in any mobile radio communication system. Channels which are particularly susceptible to noise, interference or fading (such as mobile radio communications), are thus easier to cryptographically secure using stream ciphers with no cipher feedback. Note that the GSM communication system uses a combination of both stream cipher and block data transmission, for the reasons described above.

In the same manner that the robust bit synchronization which GSM employs in order to satisfy the requirements of its speech codec also provides for the synchronization requirements of the cryptosystem it uses, the robust error detection/correction scheme provides not only for the correct operation of the speech codec, but also for the correct operation of the cryptosystem. Thus, in the GSM900/DCS1800 PLMN system, the incorporation of the cryptosystem adds no further propagation of errors than is already inherent in the block nature of the transmitted data, that is, that errors are limited to the given block in which they occur and do not propagate outside this. Moreover, the probability of such errors remaining undetected/uncorrected is low due, to the nature of the error detection/correction scheme already incorporated to ensure the proper operation of the speech codec. Note also that the intrinsic stream cipher nature of the cryptosystem results in no deviation from the essential bit-for-bit reliance of proper recovery of transmitted data on the received data being uncorrupted.

3.4.5 Signalling overheads.

All cryptosystems have particular signalling overheads associated with their operation, that is, signalling required to support the operation of the cryptosystem as distinct from the transmission of encrypted traffic. The signalling directly required to support a cryptosystem requires particular consideration as distinct from the expansion of communication traffic due to error detection/correction schemes and bit synchronization schemes, both of which are required to be implemented in any case in a digital (mobile) communication system.

The cryptosystem signalling overheads can be categorized as follows :-

- Signalling required to authenticate a communicating party.
- Signalling required to generate a cryptosystem for secure communications between parties.

The cryptographic signalling overheads may be manifest in several parts of a communications network. The most important area for consideration in mobile communications is the signalling overhead required to be accommodated by the air interface between the mobile station and the base station, since this is the most stringent design constraint with regard to the availability of channel bandwidth.

It is evident that any scheme for the authentication of communicating parties will act to reduce the overall signalling efficiency of the communication channel with regard to the efficiency of transmission of user data. Since a given security level of encryption of user data should be dependent upon the length of the cryptographic key used, then this key length can be regarded as a bare minimum overhead to be accommodated by the system concerned. Such a (secret) key would, however, not sensibly be transmitted in clear around the network concerned, and would accordingly require further signalling overheads due to the protection required to securely distribute such a key as required between the communicating parties, or to derive such a key from other security-related data, such as that used for the purpose of party authentication.

The above problem leaves the cryptosystem designer with a choice of two means of generating a cryptosystem between authenticating communicating parties. Either the secure distribution of secret keys is achieved by means of a mechanism separate from party authentication, or it is achieved as an intrinsic part of such authentication. As an example of the typical overheads for such methods of digital signature, a comparison of the respective signalling (and storage) requirements of the RSA algorithm and of the Fiat-Shamir scheme is given in Section 3.5.3.

3.4.6 Data Processing requirements.

The data processing requirements of particular cryptographic algorithms are also of paramount importance in the design of a cryptosystem. In the same manner that a restriction in channel bandwidth imposes bounds upon signalling overheads, a restriction in processing ability imposes bounds upon data processing requirements (or processing overheads). Such a restriction is encountered in the mobile station and in any intelligent token (such as a smart card) associated with the cryptosystem implementation. Thus considerations similar to the above regarding the accommodation of signalling overheads by the communication system must also be made regarding the accommodation of processing overheads by the communication system. The necessary restriction in the available processing ability is due to restrictions in processing delay, cost, physical size and in power requirement. These are of particular importance in the design of a mobile station.

As an example of the typical overheads for such methods of digital signature, a comparison of the respective processing requirements of the RSA algorithm and of the Fiat-Shamir scheme is given in Section 3.5.3.

Note that a choice of $k = 9$ and $t = 8$ for the Fiat-Shamir scheme will provide a security level = 2^{-72}. The private key could be stored in a 576-byte ROM and each signature would require 521 bytes. The average number of modular multiplications here would be 44. If the key size were doubled to 1152 bytes ($k = 18$), then the size of each signature could be reduced to 256 bytes ($t = 4$) with the security level remaining constant at 2^{-72}.

A further reduction in the number of modular multiplications is possible by optimizing the order of the multiplications to simultaneously compute the t subset products. This would reduce the number of modular multiplications to 32, which is only 4% of the number of modular multiplications required in the RSA scheme.

3.5 The Current Status of Asymmetric Algorithm Cryptosystems.

There follows an overview of the development of asymmetric algorithm cryptosystems, covering encryption schemes, authentication schemes, data integrity schemes and cryptographic key distribution schemes. Within the last fifteen years there has been a plethora of new developments in asymmetric algorithm cryptography. The examples that follow have been selected on the basis of their properties which are partially or wholly suited to the operational and security requirements of personal communication systems.

3.5.1 Current development of asymmetric algorithm encryption schemes.

Asymmetric algorithm encryption schemes have undergone development in several key areas during the last fifteen years, since their concept was first introduced by Diffie and Hellman [10]. The development of such schemes has largely fallen into two categories; those which base their security on the intractability of factorizing products of large primes (like the RSA algorithm), and those based on elliptic curves. These are broadly considered below.

3.5.1.1 Prime product cryptosystems. As shown in Section 3.2.1, the security of such ciphers is based upon the difficulty of two numerical problems; the discrete logarithm problem and the factorization problem, as stated in Section 3.2.1. In this cryptosystem, the messages, cryptograms and cryptovariables all belong to a set of integers :-

$$Z_N = \{1, \ldots, N-1\} \quad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.22$$

where $N = p \times q$
and p,q are primes

The factorization of prime products is an established mathematical problem. Given that every user of an RSA cryptosystem should have a different modulus, $n = p \times q$, for security reasons, it becomes an intractable problem to attempt to supply all users of the cryptosystem with suitable moduli. Huber [41] suggests two conditions which potential RSA moduli must satisfy if they are to be deemed suitable. The testing for the satisfaction of these conditions greatly reduces the amount of CPU time required to test potential RSA moduli with all known factorization algorithms. The two conditions are as follows :-

Condition 1 - A prime p selected as a factor of an RSA modulus n must have a large index u_p, where F_{u_p} is the smallest Fibonacci number which contains p as a factor.

Condition 2 - A prime p selected as a factor of an RSA modulus n must have a sufficiently large value of ptd(p) to prevent cryptanalytic attack. The ptd of a number n is the power-of-two-distance; a number theoretic distance function, defined as :-

$$E\{ptd(n)\} \simeq \frac{\log_2 n}{\ln \ln n} \quad \text{.........................} \quad 3.23$$

where n is random.

Sun and Hwang [42] have proposed a public-key identity-based cryptosystem which is non-interactive and is resistant to chosen-public-key attacks. There is a generally available integer N, where $N = p \times q$, the product of two large primes. If party A sends an encrypted message to party B then it computes the ciphertext C, where :-

$$C = (C_1, C_2) \quad \text{..} \quad 3.24$$

$$C_1 = g_a{}^r \text{ (MOD N)} \quad \text{..} \quad 3.25$$

$$C_2 = m \times ID_a{}^r \text{ (MOD N)} \quad \text{..} \quad 3.26$$

where $g_a = (ID_a)^{d_a}$ (MOD N) is computed by the trusted centre

and ID_a is the identity of party A.

Party B decrypts the ciphertext by solving :-

$$(C_1)^{k_a} = ID_a{}^r \text{ (MOD N)} \quad \text{..} \quad 3.27$$

note that $d_a \times k_a = 1$ (MOD ϕ(N)), as in the RSA algorithm.

and recovers the message by solving :-

$$((C_1)^{k_a})^{-1} \times C_2 \text{ (MOD N)} = m \text{ (MOD N)} \quad \text{.........................} \quad 3.28$$

The security of the system is equivalent to factoring the prime product or to solving the discrete logarithm problem.

An alternative modification to the RSA cryptosystem to convert it into an identity-based public-key cryptosystem has been proposed by Chang and Hwang [43]. This cryptosystem utilizes the properties of pair-wise relatively prime generating functions (or PRGFs). This system, however, offers no improvement to the processing overhead inherent in RSA exponentiation. Additional work in securing public key cryptographic systems against chosen-ciphertext attacks has been undertaken by Damgård [44], who presents efficient constructs for such deterministic cryptosystems.

An interesting development in the general attempt to reduce the number of modular multiplications required in a prime product-based cryptosystem has been proposed by Fiat [45]. The usual requirement for such a cryptosystem is $O(n)$ or $O(\sqrt{n})$ modular multiplications. This scheme proposes a variant which requires only $O(\log_2 n)$ modular multiplications, based on the principle of batched encryption (or signature) operations.

Several RSA hardware implementations have been achieved in recent years. In 1988 Hoornaert *et al* [46] had developed an RSA processor capable of exponentiation with 512-bit operands in less than 30 milliseconds. Their processor was a PQR6 module with a clock frequency of 14 MHz.

Gallay and Depret [47] describe a VLSI device capable of RSA exponentiation at a data rate of 512 Kbit/s for 512-bit operands, with a clock frequency of 10 MHz.

Kameyama *et al* [48] propose an RSA processor capable of exponentiating 512-bit operands at a rate of 60 Kbit/s.

3.5.1.2 Elliptic curve cryptosystems. Much work has been undertaken on the development of elliptic curve cryptosystems. These cryptosystems rely on the fact that a cyclic subgroup of the points on an elliptic curve, defined over a finite field, may be used as a one-way function, as explained by Bender and Castagnoli [49]. A brief discussion of the method of calculation in a group of points on an elliptic curve is given below. Further details of this are given in Koblitz [50].

The group of points over an arbitrary field can be defined as the set of solutions (x,y) of a certain third-order algebraic equation, including a point at infinity (∞, ∞), which is the neutral element of the group, together with an operation on these points. In a field with characteristic › 3 the general equation can be reduced by means of coordinate transformations to the form :-

$$y^2 = x^3 + ax + b \quad \ldots\ldots\ldots \quad 3.29$$

For characteristic = 3, the general equation can be reduced to :-

$$y^2 = x^3 + ax^2 + bx + c \quad \ldots\ldots\ldots \quad 3.30$$

For the characteristic = 2, the general equation can be reduced to :-

$$y^2 + y = x^3 + ax + b \quad \ldots\ldots\ldots \quad 3.31$$

Note that the condition for non-singularity is given by :-

$$4a^3 + 27b^2 \neq 0 \quad \text{..........} \quad 3.32$$

Thus the group of points of a singular elliptic curve is isomorphic to the additive or multiplicative group of the field over which the curve is defined.

Such properties may also be used to implement both signature schemes and encryption schemes, in addition to zero-knowledge identification protocols, Koyama *et al* [51].

It is vital to know if the order of the cyclic group is large enough to provide cryptographic security. The computation of the order of points on an elliptic curve over a finite field is, however, difficult in general; this discourages the use of arbitrary elliptic curves. Koblitz [52] describes a method of searching for suitable elliptic curves with random coefficients. To avoid cumbersome computation of the group order, it is suggested by Bender and Castagnoli [49] to use the subclass of elliptic curves having the coefficient a = 0 in their defining equation. These equations are then :-

$$y^2 = x^3 + b \quad \text{..........} \quad 3.33$$

for characteristic $p > 3$

$$y^2 + y = x^3 + b \quad \text{..........} \quad 3.34$$

for characteristic $p = 2$

Beth and Schaefer [53] have shown how public key cryptosystems may be constructed using multiplication on elliptic curves instead of exponentiation in finite fields. They demonstrate an attack on such a cryptosystem by embedding the elliptic curve group into the multiplicative group of a finite field via *weilpairing*; the calculation of the discrete logarithm on the curve by solving the discrete logarithm in the finite field. They illustrate an implementation over fields of characteristic p = 2.

In addressing the points for consideration in the implementation of a modification of the Diffie-Hellman key exchange protocol it is deduced that, in order to make a modification of the scheme practical, several details must be addressed including :-

- The actual algorithm for multiplication on an elliptic curve.
- The choice of parameters a and b for the elliptic curve.
- The choice of the prime modulus N.
- What data needs to be transmitted.

As shown in Koyama *et al* [51] elliptic curves may be used to implement both digital signature and public key schemes. This is based on the Trapdoor One-way Function (TOF) as proposed by Diffie and Hellman in 1976 [10]; an implementation of which was proposed by Rivest *et al* in 1978 [31]. The security relies upon the difficulty of factoring a composite number which is the product of two large primes. Similarly, other TOF schemes provide security by virtue of the difficulty of factoring and discrete logarithms, such as that proposed by El-Gamal [54]. As explained by Koblitz [50] the group E (where E are the points of an elliptic curve over a finite field) can be used to implement analogues of the Diffie-Hellman exponential key exchange scheme and the El-Gamal public key cryptosystem. Koyama proposes new TOFs based on elliptic curves over a ring. A naïve construction of a TOF based on elliptic curves over a ring Z_n is shown below.

A digital signature scheme based on $E_n(a, b)$ is constructed as follows. A signer X chooses two primes (p, q) and two parameters (a, b) such that :-

$$\gcd(4a^3 + 27b^2, n) = 1 \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.35$$

where $n = p \times q$

The signer then computes the order of the elliptic curves $E_p(a, b)$ and $E_q(a, b)$ and chooses a public encryption multiple, e, relatively prime to both $\#E_p(a, b)$ and $\#E_q(a, b)$. Note that $\#E_p(a, b)$ denotes the order (that is the number of points) of the elliptic curve $E_p(a, b)$. It is calculated as follows :-

$$\#E_p(a, b) = p + 1 + t \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.36$$

where $| t | \leq 2\sqrt{p}$ over the field Fp.
Thus all calculations are (MOD p).

The signer then computes the secret decryption multiple *d* by solving the congruence :-

$$d \equiv e^{-1} \text{ (MOD lcm}(\#E_p(a, b), \#E_q(a, b))) \qquad \ldots\ldots\ldots \quad 3.37$$

The signer then releases the parameters n, a, b and e. To sign a message M it follows that M must be associated with a point P in a public manner, where :-

$$P = (x, y) \in E_n(a, b) \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.38$$

The signer then computes the point Q = (s, t) on the curve $E_n(a, b)$ as follows :-

$$Q = (s, t) = d \times P \qquad \ldots\ldots\ldots\ldots\ldots\ldots \quad 3.39$$

The signature of the message M is the pair (s, t), which can be publicly verified by computing :-

$$P = (x, y) = e \times q \qquad \text{..} \quad 3.40$$

on the curve $E_n(a, b)$
and extracting the message M from (x, y).

Here, given a message M, then a point (x, y) on the curve $E_n(a, b)$ may be efficiently associated with M. The message M is first padded with redundancy, for example by adding zeros to form M'. Note that :-

- x is defined as the smallest integer $\geq$ M' such that $x^3 + ax + b$ is a quadratic residue (MOD n)

- y is defined as one of the square roots (MOD n) of that quadratic residue.

The security of the above proposed scheme and that of associated schemes over elliptic curves is based upon the difficulty of factoring the product n. It is suggested by Koblitz [50] that elliptic curve cryptosystems are secure if the order of the group has a prime factor of approximately 40 digits.

With regard to computational efficiency, an elliptic curve addition $P_1 + P_2$ on the curve En(a, b) requires one division, one squaring operation and one general multiplication in the ring Z_n when $P_1 \neq P_2$, and one extra squaring when $P_1 = P_2$. This results in the scheme requiring approximately (2 + c) times as much computation as the RSA scheme, where c is the number of points P on the elliptic curve $E_n(a, b)$.

The practical implementation of elliptic curve cryptosystems has been investigated by Menzes and Vanstone [55]. They discuss an integrated circuit which has been developed to perform various finite-field arithmetic operations in the field G F(2^m), where m = 593. Typical examples of such operations are multiplication in 0.065 msec and inverse in 2.5 msec, with a clock frequency of 20 MHz.

Agnew *et al* [56] describe a hardware implementation of an elliptic curve cryptosystem in the field G F(2^{593}), whereby a Motorola™ M68006 processor was used to control the operation of a custom VLSI multiplier/exponentiator. The M68006 was used to perform the manipulation and storage of points coordinates, while the VLSI device was used to calculate the inverses and the products of points coordinates.

For curves of characteristic 2 the curve $y^3 + y = x^3$ was chosen. The system calculates new points KP from a point P on the curve, where $0 < K \leq 2^n - 1$. The hybrid implementation is capable of calculating approximately 9 elliptic curve points per second for K with a Hamming weight of 30. This is approximately equivalent to a throughput rate of 5 Kbit/sec.

3.5.2 Current development of asymmetric algorithm authentication schemes.

Asymmetric algorithm authentication schemes can be divided into two main categories; those which are identification or signature schemes based on classical problems, such as the difficulty of factorizing the product of two large primes and those which are constructed to be zero-knowledge. A zero-knowledge scheme is one in which an authenticator (either an identifier or a signer) sends a set of apparently random bits to a verifier thus imparting no knowledge to the verifier yet nevertheless permitting the verification of the authenticator.

3.5.2.1 General identification and signature schemes. Chang and Lin [57] have proposed a new identity-based signature scheme, where the signature is dependent both upon the identity of the signer and upon the signed message. It has the following properties :-

- No requirement for the changing of public or private keys.
- No requirement for the maintenance of a key directory.
- No requirement for the services of a third party.

The scheme is based upon Rabin's public key cryptosystem [58]. The signature procedure of the sender and the verification procedure of the receiver can both be achieved by several multiplication and addition operations. The signature protocol has two phases; the first phase is where a trusted centre chooses two large primes, and the second phase involves the signature generation and verification. Both the signature generation procedure of the sender and the verification procedure of the receiver can be achieved efficiently by several multiplication and addition operations. The security of the scheme depends upon the ability to solve a quadratic congruence equation, since the individual large primes are not known. This is as difficult as factorizing the product of these primes. This scheme has similar properties to those of the RSA scheme, plus it is identity-based. There is no requirement for any exponentiation operations.

Desmedt and Frankel [59] have proposed a scheme for the shared generation of authenticators and signatures in the form of a threshold scheme, in which l individuals are given shares such that a subgroup $k \leq l$ are needed to generate a signature (authenticator), but a sub-group $< k$ cannot. When k people have finished signing, no party may then successfully perform an impersonation attack. They propose threshold schemes for RSA signatures, provably secure authenticators and for unconditionally secure authenticators.

Progress in undeniably secure authenticators has also been achieved by Chaum *et al* [60], who propose undeniable schemes where signers are unconditionally secure. Their schemes also offer an improvement in efficiency over many other unconditionally secure signature schemes. Their work also includes efficient collision-free hash functions and zero-knowledge proofs for arithmetic formulæ.

Similar work has been achieved by Chaum and Roijakkers [61], who propose an unconditionally secure digital signature in the form of a polynomial time protocol in which the signer can convince a verifier that a signature is valid. Furthermore, any such convinced verifier can subsequently convince another party of the authenticity of the original signature, without any interaction with the original signer.

Interactive, probabilistic public-key encryption protocols have been proposed by Galil *et al* [62], in which the sender and receiver of a message, in addition to the message itself, can be authenticated. Their scheme is secure against any feasible attack by a participant, including a chosen-plaintext attack.

Ohta and Koyama [63] have analysed the intrinsic security against forged signatures afforded by mixed-type digital signature schemes, and have proposed a meet-in-the-middle attack which takes less time than an exhaustive, brute-force attack. They demonstrate how secure mixed-type digital signature schemes may be constructed, and show how to ensure that the security of such schemes is mainly dependent upon the complexity of factoring rather than on the complexity of the meet-in-the-middle-attack. This is achieved by ensuring that the domain size of the signature function is greater than a particular minimum size. This minimum size is demonstrated to be 500 bits for the RSA cryptosystem.

Fujioka *et al* [64] have proposed a digital signature algorithm with a computation speed of at least twenty times that of the RSA scheme, but with a comparable key length and signature length. They present a software implementation on an 8-bit micro-processor smart card, which is capable of signature generation in approximately 0.2 seconds when operating at 5 MHz. Their scheme requires a maximum of 3K bytes for each of the program and the data. Refer to Table 3.3.

3.5.2.2 Zero-knowledge schemes. The Fiat-Shamir zero-knowledge cryptosystem for identification and authentication, as detailed in Section 3.3, has proved to be as much of a landmark in cryptography as was the advent of the RSA scheme. The work of Fiat and Shamir has been heavily scrutinized and built-upon.

The probability of forgery in the Fiat-Shamir scheme is given by 2^{-kt} and is an absolute constant. This means that there is no need to cater for future technological or cryptographic developments; only the processing overhead need be considered in this respect. Moreover, a security level of 2^{-20} or 2^{-30} is more than adequate if it means that an attacker must make 1,000 representations each day to even stand one chance of success every 3,000 years. A security level of 2^{-20} may be achieved by selecting the values of $k = 5$ and $t = 4$. This results in 14 modular multiplications being required to generate or verify a proof of identity. The number of communication bytes exchanged by each party during every proof of identity is then 323, and the secret S_j values can be stored in 320 bytes of ROM.

The resources of the system (the processing and communication overheads of the scheme and the security level it affords) can be traded off in several ways by varying the values of k, t and the number of 1's in each e_{ij} vector. It is the relative value of these resources that will determine the actual values chosen. Note that, unlike the RSA scheme, users may pipeline their computation and use parallel multipliers to give a further reduction in processing time. Refer to Chapter Six for further details of the way in which zero-knowledge schemes may be optimized to suit the particular system in which they are used.

An alternative to the Fiat-Shamir scheme has been proposed by Stern [65]. The identification protocols proposed are based on simple operations such as multiplication by a fixed matrix over a two-element field. Such a matrix may be viewed as the parity-check matrix of a linear binary error-correcting code. The communications complexity of the protocol is marginally worse than in the Fiat-Shamir scheme, but the operations required to be performed are very simple. The real problem with the proposed scheme is the amount of memory required to store the matrix and the words, which is typically 630 Kbits.

Ohta and Okamoto [66] have proposed an improvement to the Fiat-Shamir scheme that addresses a particular problem inherent in the scheme; namely that the transmitted information size and the stored information size cannot simultaneously be small. Their scheme is based upon the difficulty of extracting the L-th roots (MOD n) when the factors of n are unknown. The speed of their scheme is at least one order of magnitude faster than the RSA scheme, but it is relatively slow in comparison with the Fiat-Shamir scheme (refer to Table 3.3). It is capable of being implemented as a signature scheme or as an N-party authentication scheme.

Micali and Shamir have proposed another improvement to the Fiat-Shamir scheme for zero-knowledge based identification and digital signature, which reduces the verifier's complexity from between 10 and 30 modular multiplications to <2 modular multiplications, and leaves the verifier's complexity unchanged.

Brandt *et al* [67] have developed a zero-knowledge interactive proof system of knowledge, being modified into an authentication scheme with secret key exchange for subsequent conventional encryption. They maintain that their whole protocol, which entails the mutual identification of two users, the exchange of a random common secret key and the verification of verification of certificates for the public keys (RSA, 512 bits), takes less than 0.7 seconds when implemented on a standard 32-bit processor such as the Intel 80386 or the Motorola™ 68030.

Beth [68] has proposed a scheme similar to that of Fiat and Shamir, but based on the El-Gamal scheme [54] for key exchange, for public key encryption and for digital signature. It represents a simplification of the Fiat-Shamir scheme where, for only one signature and a small number of iterations, the storage of the signature s_j and the exponentiated logarithm r requires approximately 64 bytes each. This reduced storage size and processing requirement make the protocol suitable for smart card implementation.

Knobloch [69] has proposed a smart card implementation of the Fiat-Shamir identification scheme. The implementation allows the authentication of a 120-byte identification string at a security level of 2^{-20} within an average time of 6 seconds. The implementation proposed in [69] requires 256 bytes of RAM and 2K bytes of EEPROM.

Guillou and Quisquater [70] have also proposed a practical zero-knowledge protocol suitable for smart card implementation, which minimizes both transmission and memory requirements. Their proposed scheme, however, requires an increased number of modular multiplications compared to the Fiat-Shamir scheme. Refer to Table 3.3.

Schnorr [71] has proposed another scheme for efficient identification and signature for smart cards based upon discrete logarithms. The scheme utilizes pre-processing of the exponentiation of random numbers. A typical signature length is about 212 bits and signature generation requires about 12 modular multiplications. Refer to Table 3.3. The security of the Schnorr scheme has been analysed by de Rooij [72] who shows that, by virtue of the pre-processing algorithm, the Schnorr scheme leaks too much information, resulting in a significantly reduced security level to that originally claimed. Refer to Table 3.3.

Shamir [73] has proposed an interactive identification scheme based on zero-knowledge proofs, by virtue of the concept of the permuted kernel problem (PKP). The PKP is defined as follows. Given a m $\times$ n matrix A, a n-vector V and a prime p, find a permutation π such that :-

$$V_{\pi} \in K(A) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 3.41$$

To use the PKP as an identification scheme, the parties agree on a universal matrix A and prime p and then each party chooses a random permutation π (which serves as their secret key) and a random vector V (which serves as their public key) in accordance with equation 3.41. Users can now establish their identity by proving their knowledge of the secret permutation π. By using zero-knowledge proofs, provers can guarantee that eavesdroppers and dishonest verifiers will learn nothing about π which may later enable them to misrepresent themselves as the prover to other parties. Typical specifications of the PKP identification scheme are given in Table 3.3, below.

3.5.3 Current development of asymmetric algorithm data integrity schemes.

Data integrity schemes *per se* have undergone relatively little development in recent years with regard to asymmetric algorithm cryptography. Emphasis has been put on schemes which provide inherent protection for the integrity of data by means of digital signature, usually as a part of several security functions of a particular cryptographic scheme. An example of one of the few published works regarding data integrity by means of asymmetric algorithm cryptography is that of Haber and Stornetta [74]. Their proposed solutions to data integrity involve the use of time-stamping mechanisms via digital signature and collision-free hash functions and incorporate such measures as linking and distributed trust. Linking is a technique of incorporating data from previous authentication requests into signature vectors (or certificates) in order to establish that the time-stamp occurred after those requests. Distributed trust is a technique of choosing cooperating parties on a (pseudo)random basis to time-stamp a message and thus to act as "witnesses" to the authenticity of a message. The security of this technique is inherent in the small probability of all chosen authenticating parties being corrupt.

security scheme	security level	number of prover modular multiplications	number of verifier modular multiplications	total storage bits	communication bytes
Rivest *et al* (RSA)	$e^{\sqrt{\ln N \times \ln(\ln N)}}$	$\frac{3\lvert n\rvert}{2}$	≥ 2	$\lvert n\rvert$	$2\lvert n\rvert$
Fiat-Shamir	$2^{-k.t}$	$\frac{t(k+2)}{2}$	$\frac{t(k+2)}{2}$	$k.\lvert n\rvert$	$k.t + (2\lvert n\rvert.t)$ (bits)
Guillou-Quisquater	$2^{-\mathrm{Id}(c)}$	$\simeq 3 \times \mathrm{Id}(c)$	$\simeq 1.5 \times \mathrm{Id}(c)$	$\lvert n\rvert$	$\frac{\lvert n\rvert}{4} + \mathrm{Id}(c)$
Schnorr	$\leq 2k^{4(d-2)}$	$\frac{3 \times \mathrm{Id}(q)}{2}$	$\frac{3 \times \mathrm{Id}(q+t)}{2}$	$\leq 1k$	$\simeq \frac{t+q+p}{8}$
Shamir	2^{-t}	256 single-byte multiplications	256 single-byte multiplications	160	500 bits
Ohta-Okamota	L^{-tk}	$\frac{5l+1}{2}$ $(L = 2^l)$	$\frac{5l+1}{2}$ $(L = 2^l)$	$\lvert n\rvert$	$2\lvert n\rvert + L$ (bits)
Fujioka *et al*	$e^{\sqrt{\ln N \times \ln(\ln N)}}$	†	†	†	$2\lvert n\rvert$

(† Complex evaluation: No general equation, but are bounded by choice of parameters)

Table 3.3 A Comparison of the Specifications of Zero-Knowledge Schemes

3.5.4 Current status of asymmetric algorithm key distribution schemes.

Any cryptographic algorithm may be used to distribute cryptographic keys by virtue of the security provided by that particular algorithm. There has been, however, a significant amount of development in the last ten years regarding mechanisms for the secure distribution of cryptographic keys by means of asymmetric algorithm cryptography. Such schemes offer intrinsic advantages such as there being no requirement for any secret information to have been previously shared between parties which are to share a cryptographic key, and the ability to create an identity-based key-sharing scheme to ensure the authenticity (and integrity) of any distributed keys.

As discussed in Section 3.5.4, there has been a proposal for a modified Diffie-Hellman protocol using elliptic curves. It has been shown how the proposed analogue of the Diffie-Hellman protocol for exponential exchange of cryptographic keys appears to be immune from certain attacks.

Identity-based cryptosystems can simplify key management in cryptosystems. The work of Fiat and Shamir has already been discussed; Okamoto [75] has proposed an identity-based scheme as a development of the Diffie-Hellman public key distribution system. In these schemes for two parties, messages between parties are authenticated using each party's identification information. If two or more parties wish to hold a conference they must derive one secret communication key for each link in the network. This common key is called a conference key. Koyama and Ohta [76] have proposed an identity-based cryptosystem for generating a conference key with authentication; an identity-based conference key distribution system. Their cryptosystem has protocols for three configurations; ring, complete graph and star. The security of their cryptosystem is based upon the classical difficulty of factoring a composite number (prime product) and of computing the discrete logarithm. It is thus analogous to the RSA cryptosystem.

Tatebayashi *et al* [77] have proposed a key distribution protocol which has been specifically selected for suitability to the hardware limitations of mobile terminals in mobile communication systems. Their solution takes into consideration the intrinsic problems associated with centralized key distribution systems, particularly the processing requirement of a centralized key distribution facility. This is a particular problem when using public key cryptography. A similar problem of computational complexity exists with the Diffie-Hellman protocol for the exponential exchange of cryptographic keys, where computation in a high-order finite field is necessarily processor intensive. Their solution involves the distribution of the processing of an RSA-based protocol between a mobile terminal and the host network in such a manner that the processing at the mobile terminal is minimized. Their protocol combines three special security mechanisms: a structure in the transmitted data to prevent an attacker from utilizing the distribution property of RSA-related cryptography, a time stamp mechanism to prevent replay and an identity verification mechanism to prevent masquerading.

Yacobi and Shmuely [78] have investigated the application of zero-knowledge techniques to the problems associated with the distribution of cryptographic keys. They assert that, although such techniques are less suited to key distribution than they are to identification and to signature, they are nevertheless usable for the purposes of key distribution with certain relaxed criteria. Their solution is minimum-knowledge, being a variation of the Diffie-Hellman protocol. Its security is equivalent to RSA but it runs faster. They allow the general proof of security of key distribution protocols from attack by a malicious adversary.

Günther [79] has proposed an identity-based approach to the problem of key distribution in large networks. Each new communicating party is required to exchange communications with a key authentication centre only once and is then able to exchange authenticated keys with each user of the network. The work is a development of that of Rivest, Shamir and Adleman who indicated a solution to the problem of authenticating public keys. A key authentication centre signs the public key of each user and thereby guarantees its authenticity. The implementation of this solution for the authentication of the public keys used in the Diffie-Hellman scheme is typically not very practical. Shamir [80] has proposed an interesting approach to identification and signature where the user of some communications facility only needs to know the "name" of a communications partner and the public key of the key authentication centre. Such schemes are correspondingly called "identity-based". Günther has adapted the approach to create a cryptosystem where two parties construct keys which agree if they are both legitimate and conform to the protocol. The identity-based key exchange protocol has three phases; set-up, pre-authentication and authentication. The first two phases ensure that parties are verified and are then given a secret signature of their name by the key authentication centre. The central property of identity-based protocols is that after the completion of the pre-authentication phase a party is able to authenticate itself (authentication phase) to any other user without uncovering the secret signature of its name. The operations involved in the protocol are identical to those involved in the Diffie-Hellman key exchange protocol, and the security level depends only upon the length of the words exchanged and not upon the number of exchanges.

Davida *et al* [81] have proposed a protocol for a key distribution system over an insecure, but authenticated, channel using a one-way function. Their scheme requires $O(n)$ time for agreement on a number in the range $1..n^2$; an attacker requires $O(n^2)$ time to obtain the secret key. Thus the number of steps required for an attacker to cryptanalyse the cryptosystem is the square of the number of steps in the protocol. Moreover, if pre-computation is permitted for one of the parties in the key exchange and also for an attacker, then this performance can be improved significantly; typically resulting in an attacker requiring the fourth power of the number of steps in the protocol to cryptanalyse the cryptosystem. Their work illustrates the requirement that a protocol is secure if and only if, when the legitimate communicating parties use resources in an amount N, the resources necessary to break the protocol are an exponential function of N.

A scheme has been proposed by Ferrer and Rotger [82] for a fully secure key exchange protocol, which provides secrecy and authentication, with no previously shared secrets and with a moderate amount of computation. This mode of operation can be succeeded by a second mode, where an exchanged key can be subsequently used either for the encryption or decryption of messages, or it may be used to sign them. Their proposed system requires that several parameters be made public by publishing them via a suitable channel.

Tsujii and Chao [83] have proposed an identity-based key sharing cryptosystem which is non-interactive. This obviates the requirement for preliminary communication. Their proposed scheme also provides high security against conspiracy of parties, to which non-interactive schemes are susceptible. Such a conspiracy would possibly enable a number of parties to share the individual secrets given to them by the trusted centre, resulting in disclosure of the secret information contained in the centre or the ability to forge the common secret keys between some other communicating parties. Their scheme, however, requires relatively complex computation which would not be suitable for implementation in a mobile station.

Maurer and Yacobi [84] have proposed a non-interactive public key distribution system which is based on a "trapdoor" one-way function which allows a trusted authority to compute the discrete logarithm of a given number modulo a publicly-known composite number *m*. This computation, however, is unfeasible for an adversary not knowing the factorization of *m*. Without interaction with either a key distribution centre or with the recipient of a given message, a party can generate a mutual secret cipher key based solely on the identity of the recipient and on the secret key of the party. The message may then be encrypted with the generated cipher key and sent to the recipient over an insecure channel. Unlike most other identity-based key distribution schemes, this protocol has no requirement for the previous exchange of public keys, their certificates or any other information. This allows the implementation of the exchange mechanism with a minimal exchange protocol. The security of the protocol is based upon the classical difficulty of factorizing a prime product and on solving discrete logarithms over a finite field.

Fumy and Munzert [85] have undertaken a modular approach to key distribution and have characterized the various properties associated with the plethora of key distribution mechanisms. They characterize key distribution mechanisms by six generic requirements, namely :-

- Data confidentiality.
- Modification detection.
- Replay detection / timeliness.
- Authentication.
- Data origin authentication.
- Proof of delivery.

Their analysis of key distribution mechanisms is made with the aid of building blocks and composite rules, and transforms the cryptographic protocols into a generic form. This allows the protocols to be analysed with regard to the results the respective protocols achieve and on which assumptions they depend, which is useful for the analysis and the construction of such protocols.

3.6 Conclusions

This chapter has dealt with the properties of asymmetric algorithms and public key cryptography. Their intrinsic properties have been highlighted, particularly those of statistical re-distribution and error propagation.

The main emphasis of this chapter has been an exploration of the possible applications of asymmetric algorithms and public key cryptography to communication systems. The manner in which a public key cipher may be used to generate a digital signature scheme has been demonstrated, as has how it may also be used to generate a cryptographic key distribution scheme in relation to the limitations inherent in the use of private key (symmetric algorithm) cryptography.

The importance of the ability to generate both random numbers and prime numbers in such cryptographic schemes has been highlighted and the RSA public key algorithm has been introduced as a typical example of such a cryptosystem, this being the generally-accepted bench-mark in such cryptography. The mathematical basis of the RSA algorithm has been explored in depth and considerations have been made regarding its practical implementation.

The Fiat-Shamir scheme for zero-knowledge authentication and digital signature has been presented in comparison to the RSA scheme. Its mathematical basis has been explored in depth and a practical example of its implementation has been presented, together with various considerations regarding such practical implementation of the Fiat-Shamir scheme.

The fundamental issues regarding the practical incorporation of any cryptographic scheme into a real communication system have been explored in depth. The security level provided by various cryptosystems has been quantified and has been shown to be dependent upon certain parameters intrinsic to the given cryptographic algorithm.

Cryptographic key management has been shown to be a major factor for consideration in the implementation of any cryptosystem. It has been shown that such key management can be broadly sub-divided into four essential categories, namely :-

- Random number generation.
- Prime number generation.
- Elimination of weak keys.
- Key distribution.

Cryptosynchronization has also been shown to be a crucial factor for consideration in the incorporation of any cryptographic scheme into a communication system, together with error propagation, and signalling and processing overheads.

It has been shown that digital (mobile) communication systems, which typically possess a robust synchronization scheme by virtue of their intrinsic requirement for bit synchrony, will also cater for the requirements of cryptosynchronization.

Similarly, it has been shown that such communication systems will typically provide a mechanism for error detection/correction, which may also help improve the bit error performance of an incorporated cryptosystem. It is expected that any cryptosystem which is to be incorporated in a communication system with a mobile channel will accordingly utilize a self-synchronizing stream cipher for reasons of efficiency and resilience against corruption by errors caused by the channel propagation characteristics; refer to Proctor [86]. The design of self-synchronizing stream ciphers is explained by Daemen *et al* [87].

The signalling and processing overheads have been shown to be a crucial factor for consideration as to whether a particular cryptographic algorithm is suitable for incorporation into a particular communication system. This is of particular importance in a mobile radio system where channel bandwidth and processing power are strictly limited. It has also been shown that the cryptographic protocol associated with a given cryptographic algorithm is fundamental to the consideration of the overall signalling overhead.

The current status of asymmetric algorithm cryptosystems has been explored, with consideration being given to four areas :-

- Asymmetric algorithm encryption schemes.
- Asymmetric algorithm authentication schemes.
- Asymmetric algorithm data integrity schemes.
- Asymmetric algorithm key distribution schemes.

It has been shown how zero-knowledge schemes provide both functional and processing / signalling overhead advantages in comparison with general asymmetric algorithm encryption / authentication schemes. Particular significance has been shown in the Fiat-Shamir zero-knowledge scheme, with reference to several proposed modifications such as the Guillou-Quisquater zero-knowledge scheme.

Data integrity schemes using asymmetric algorithm cryptography have been found to have undergone very little development in relation to the other applications of such cryptography.

It has also been found that key distribution schemes using asymmetric algorithm cryptography are largely based upon the Diffie-Hellman scheme for exponential key exchange; zero-knowledge schemes being intrinsically unsuitable for this purpose.

The general results of the investigation into asymmetric algorithm cryptography, as summarized in this chapter, are used in the application of security to the current mobile communication networks. Refer to Chapter Five and Chapter Six for a detailed appraisal of the application of such cryptographic techniques to mobile communication networks.

CHAPTER 4. FORMAL ANALYSIS OF THE SECURITY LEVEL PROVIDED BY CRYPTOGRAPHIC PROTOCOLS.

This chapter presents an overview of various formal logical methodologies which may be applied to the specification, implementation and verification analysis of cryptographic key distribution and authentication protocols. It builds upon the theoretical principles of cryptography as formulated in Chapter Two and Chapter Three, to demonstrate how a complete cryptosystem may be designed and evaluated.

The use of such formal methodologies is examined in its relation to the analysis of security goals and security mechanisms, and also in its application to the management of complexity.

The security functionality afforded by the use of authentication protocols which incorporate an intelligent token is examined. The use of trust relationships in the evaluation of security functionality is analysed and the concept of the Cryptographic Finite State Machine (CFSM) is explored.

4.1 Introduction

All cryptosystems provide security functionality by means of data being communicated with particular cryptographic algorithms via their respective protocols, such functionality being embedded within one or more layers of a given communication system. This results in a security architecture existing within the network architecture of the communication system concerned. The development of the cryptographic algorithms may often be assisted with the use of formal methods of specification, requirements analysis and system design, verification and validation, as an overall aid to system analysis and evaluation.

4.1.1 Security architecture

All cryptosystems, when embedded within a communications network, are bounded by the architecture of that communications network or, conversely, impose bounds upon the nature of the architecture of that communications network. The nature of the above bounds is principally dependent upon the security functionality to be provided, but it may also be dependent upon the nature of the cryptosystem chosen to provide the desired security functionality. This results in a logical connectivity, defining which entities in the communications network are required to communicate with each other.

The architectures of intelligent networks in communication systems are becoming standardized with the effect that the incorporation or modification of security mechanisms is essentially a matter of the evaluation of the properties of the particular cryptographic protocols and of the effects of the incorporation of such protocols in a particular network architecture. In view of this there has been much development of the techniques for the analysis and the development of security-related protocols, including the inter-relationship between such protocols and the particular network architecture. Certain analysis techniques are able to indicate the effect of security-related protocols on the architecture of the host communications network. This approach has been chosen for the purposes of formal analysis of network architecture. A selection of the above protocol analysis/development techniques is considered below.

4.1.2 Formal analysis of cryptographic protocols using BAN-type logic

There has been significant progress in the application of formal methods to cryptographic protocols. Berger *et al* [88] define a cryptographic protocol as a mathematical object and security as a property of that object. Their work shows how the concatenation of two, independently secure, protocols can result in an overall insecure protocol. An example of this is the concatenation of Rabin's encryption function [89], using two different composite numbers. Their work also shows how the use of RSA with small exponents exhibits similar problems.

Hauser and Lee [83] have developed the widely accepted work of Burrows, Abadi and Needham [91] regarding the BAN formal analysis of authentication protocols. Their resultant development tool is based upon two building blocks; an analytical protocol verification part employing new authentication logic, and a simulation and modelling part based on Cryptographic Finite State Machine (CFSM) definitions of the participating principals, as previously introduced in [27].

The authentication protocol proposed by Needham and Schroeder [28] has been popularly accepted and has been incorporated into such cryptosystems as Kerberos, refer to Steiner *et al* [92]. Its security is by virtue of the fact that mutual trust in the identity of the communicating parties can be established by exchanging an authenticating secret vector over a possibly un-trusted network. This security is dependent upon the assumption that the employed encryption is strong.The protocol was exposed to contain weaknesses, as discussed in Denning and Sacco [93] and Bauer *et al* [94]. The problems caused by such weaknesses in authentication protocols were addressed by the BAN approach [91], a full description of this logic and its application is presented in the latter reference. The BAN logic is defined as follows :-

"Assumptions about the beliefs of an initial state are transformed by a small set of logical postulates to beliefs in a final state. The conclusions of the final state must at least include each party's belief in a shared session key or a public key. Within this chain of the applications of the logical postulates, each message of the protocol monotonically leads to a new, intermediate state of beliefs."

Burrows *et al* demonstrate that a key distribution protocol (between parties *A* and *B*) is correct if and only if the following conditions are satisfied after parsing the protocol :-

• Both *A* and *B* believe *CK* (the session key) is a secret shared exclusively between *A* and *B*. Syverson [95] refers to this as a first-level belief.

• Both *A* and *B* believe that the other has the first-level belief. Syverson refers to this as a second-level belief. If a party holds a second-level belief, then it believes that a secure channel has been established by the key distribution protocol.

• The causal relationship between the first-level beliefs and the second-level beliefs holds, *i.e.* if *A* gains the belief "*A* believes *B* believes *CK* is shared between *A* and *B*" at time t, then *B* must have gained the belief "*B* believes *CK* is shared between *A* and *B*" at time $t' \leq t$.

• *CK* should be distributed exclusively to *A* and *B*; thus no parties other than *A* and *B* should have beliefs about *CK*.

The BAN analysis relies upon an idealization procedure which is awkward to use because its rules are not transparent. The improved logic aims to provide protocol designers with a better basic development tool and an easy means of illustrating the protocols. Syverson also explains why the (undeveloped) BAN approach is not suitable for the analysis of protocol security.

The BAN analysis tool supports three fundamental types of cryptosystem, namely :-

• Symmetric algorithms using secret keys such as the DEA.
• Asymmetric algorithms using public/private keys such as RSA.
• Schemes employing modulo-2 addition with stream ciphers such the Vernam cipher.

The simulation part of the tool is not necessary to obtain the verification proofs of a protocol. The protocol designer is able to assess the functionality of a protocol based upon the *seeing* and *belief* mechanisms of the logic. Three concepts can be simulated with the tool, namely Cryptographic Finite State Machines (CFSMs), active intruders and time. The tool has been implemented in Prolog™ and has been successfully tested on many protocols, including several BAN examples. A CFSM model for the hardware implementation of cryptographic algorithms has also been developed; refer to Daemen *et al* [96] .

BAN-type logic has also been developed for the study of authentication and delegated trust in a system incorporating smart cards, Abadi *et al* [97]. Their work describes a range of public-key smart card authentication protocols, allowing various compromises between cost and security level. The algorithms explored demonstrate how public-key cryptography is an appropriate choice for authentication and delegation of trust in distributed systems, these algorithms being part of RSA-based cryptosystems and zero-knowledge cryptosystems. Their work concentrates on proving the security level of smart card protocols, whilst reducing the requirements placed upon the smart card. The concept of an on-line trusted centre is introduced into their proposed cryptosystem, for purposes including the removal of any requirement for the smart card to generate time stamps and to verify certificates.

The BAN approach to protocol analysis, however, is not without its own flaws. Snekkenes [98] has shown how the BAN protocol annotation rules are erroneous due solely to protocol step permutations that are undetectable to the BAN logic. An important finding of Snekkenes' work is that there is unlikely to be a suitable extension of BAN logic for the analysis of zero-knowledge protocols, but that the BAN approach may be successfully developed to incorporate an extra termination proof obligation and consequently be used to detect a broader class of security flaws.

Snekkenes has also shown how the rôles of the principals must be clearly defined in cryptographic key distribution protocols [99], in order that the security of such protocols may be properly assessed using formal techniques. He demonstrates how formal techniques such as BAN analysis can fail to identify security flaws in certain protocols, where a principal may assume more than one rôle such as a session key provider and a session key user.

4.2 Alternative Formal Methodologies for the Development and Assessment of Cryptographic Protocols

The essential aim of formal methods is to verify the assumptions on which a given cryptographic protocol provides security and to verify that the protocol achieves the design goals whilst maintaining the desired level of security. Formal methods, however, typically fail to meet the requirements for the design and analysis of security architectures, since they are often overly simplistic or are so algorithm/protocol specific that they tend towards a proof of the security of a particular cryptosystem, rather than be a general development or assessment tool for the aid of the cryptosystem designer.

Meadows [100] has described a formal specification language and verification technique for analysing key management protocols. The work is comprehensive in its consideration of the relevant aspects regarding protocol specification and verification, but it is still in an early stage of evolution and requires substantial further development before it can be applied universally and simplistically to cryptographic protocols in general.

Glasgow *et al* [101] have presented a formal framework called Security Logic (SL) which has been developed for specifying and reasoning about security policies, and for verifying that the system design adheres to such policies. Permission is used to specify secrecy policies and obligation is used to specify integrity policies. A security policy is given as a set of policy constraints on the SL model. The combination of policies is addressed. Their work, however, concentrates on the security of asynchronous distributed systems, where it represents a formal proof system for the verification of implementations of a particular security policy, rather than on the use of knowledge and belief to analyse the security of authentication protocols and encryption-based key distribution protocols.

Syverson [95] has proposed a logic and associated formal semantics specifically designed to represent and analyse cryptographic protocols. His work is based on a set of rules and axioms for quantified epistemic logic (logic of knowledge) of encryption, rather than doxastic logic (logic of belief). The Needham-Schroeder key distribution protocol [28] is analysed as an example, where the security is shown to be based on the logic of nonces, (refer to Section 4.4.2). Potential weaknesses in the protocol are exposed. The logic is yet to be extended to capture temporal features; particularly useful when describing inherently temporal features such as nonces.

Rueppel [102] has described a formal methodology to aid the system designer in the assessment of the options available at a given layer of the architecture of a secure system, in the minimization of the complexity of the system and in the indication of when the security architecture of a given cryptosystem is complete. An extract from his methodology is given in this Section and examples of its application are given in Section 4.3.

There are two fundamental security goals, on which all other security goals can be considered to be based :-

- Data confidentiality: This is the property that whoever may receive a message, only a legitimate party will be able to read it.
- Data authenticity: This is the property that whoever may receive a message, only a legitimate party will be able to write it.

Other security goals which can be considered to be based upon the above fundamental security goals include :-

- Message non-repudiation (origin authentication).
- Party (peer-entity) authentication.
- Data integrity.

4.2.1 Fundamental Logic

Rueppel [102] defines a logic to allow the construction of cryptosystems. An example of the symmetric encryption over a channel is shown in production 4.1 below :-

$$Secu(N^2) \rightarrow sym(secu, N^2), Secu(N^2) \quad \dots\dots\dots\dots \quad 4.1$$

where $Secu(N^2)$ LHS $\equiv$ goal of securing N^2 (data) channels
$sym(secu, N^2)$ $\equiv$ symmetric cryptosystem
$Secu(N^2)$ RHS $\equiv$ goal of securing N^2 (key) channels

In the above example the security over the symmetric channel can only be maintained if the cryptovariable is securely transmitted. This implies that the security of the data link implies security of the cryptovariable link. It follows that achievement of the LHS security goal implies achievement of the RHS security goal(s), and that when replacing lower level security goals by higher level security goals in the construction of a security architecture, all security goals must be resolved during the construction in order to achieve the original security goal.

4.2.2 Evolution of a security architecture

Rueppel illustrates how a system may be designed to provide homogeneous authentication for N parties over a symmetric channel, as described by productions 4.2 to 4.5 thus :-

$$Auth(N) \rightarrow sym(secu, N^2), Secu(N^2) \quad \dots\dots\dots\dots \quad 4.2$$
$$Secu(N^2) \rightarrow asym(secu, N), Auth(N) \quad \dots\dots\dots\dots \quad 4.3$$
$$Auth(N) \rightarrow phys(auth, N), ta(auth, N), asym(auth, 1), Auth(1) \quad \dots\dots \quad 4.4$$
$$Auth(1) \rightarrow phys(auth, 1) \quad \dots\dots\dots\dots \quad 4.5$$

Production 4.2 shows a choice of satisfying the original security goal *auth(N)* by means of a symmetric channel. It also shows how the complexity of the problem increases from N to N^2, since there is one key for each possible pair of communicating parties. This means the security goal is $Secu(N^2)$ in order to securely distribute N^2 keys.

Production 4.3 shows a choice of using a public key cryptosystem to distribute the keys as above. This has a security goal of *Auth(N)*, which here reduces the complexity from N^2 to N. Note that this is not the same case as the original security goal, since public keys do not change as frequently as the data in the symmetric channel link.

Production 4.4 shows the inclusion of a trusted centre *ta* to certify the public keys. Access to the trusted centre is via physically secure channels, and would typically be used to send public keys to the trusted centre from each communicating party, to be followed by an asymmetrically signed response from the trusted centre to each communicating party. This response is a certified copy of the respective public key sent to it by a particular communicating party. This constitutes a certificate. Each such communicating party will require a copy of the public key of the trusted centre in order to verify the authenticity of the certificate it receives from the trusted centre, as represented by the security goal *Auth(1)*. The important aspect here is that the security management complexity has been reduced from N to 1 by means of the use of a trusted centre, and that such a trusted centre is not required to be trusted with the individual secret data of the communicating parties.

Production 4.5 shows the requirement for authentic physical distribution of the public key of the trusted centre. This could be achieved by physical publication. The final security goal has now been achieved and thus the security architecture is now complete.

4.2.3 Analysis of a security architecture

An example of the analysis of a security architecture is as follows :-

1. In order for two parties A and B to generate a cryptosystem for secure communications between them they each select their own secret keys k_A and k_B.

2. Both parties then encrypt their secret keys by means of a public encryption key P_{KMC} belonging to a trusted key management centre, KMC. The resultant cryptograms are then sent to the centre :-

$A \rightarrow KMC: \ P_{KMC}(k_A)$ 4.6

$B \rightarrow KMC: \ P_{KMC}(k_B)$ 4.7

3. The KMC then recovers k_A and k_B by decrypting the cryptograms using its secret decryption key S_{KMC}. The KMC then selects a session key k_s for the parties A and B, transmitting k_s to them via the symmetric channels established by k_A and k_B :-

$KMC \rightarrow A: \ E(k_s)$ 4.8

$KMC \rightarrow B: \ E(k_s)$ 4.9

4. The parties A and B then recover the session key k_s by means of their independent secret keys k_A and k_B, thus creating a private symmetric cryptosystem between them. Note that the centre KMC will continually broadcast its public encryption key P_{KMC} to all N parties.

Using the logic defined above in Section 4.2.1, this protocol may be translated into the following security architecture :-

$Secu(N^2) \rightarrow sym(secu, N^2), Secu(N^2)$ 4.10

$Secu(N^2) \rightarrow sym(secu, N), ta(secu), Secu(N)$ 4.11

$Secu(N) \rightarrow asym(conf, 1), Auth(1)$ 4.12

$Auth(1) \rightarrow conf(1)$ 4.13

Upon parsing this architecture, it can be seen that it is impossible to satisfy production 4.12, since it is impossible to achieve *Secu (N)* with *asym (conf, 1)*. This is because any party in possession of the public key P_{KMC} can impersonate any other party in the system. This results in the above security architecture achieving neither confidentiality nor authenticity.

4.2.4 Security goals

The security goals in a system may be either homogeneous or non-homogeneous; the former is where all communicating parties within a given system have identical capabilities and privileges, the latter is where there is a differential in such party capabilities and privileges. It is also necessary to distinguish between distributed and centralized security goals, since both are a factor in many multi-party communication systems. These are explained below.

4.2.4.1 Distributed security goals. There are three distributed security goals, namely :-

- *Conf (N)* which denotes homogeneous individual confidentiality. This is the channel from which the party can read, but every other party can only write.
- *Auth (N)* which denotes homogeneous individual authenticity. This is the channel to which the party can write, but every other party can only read.
- *Secu* (N^2) which denotes homogeneous individual security. This is the secure channel which exists between every pair of parties.

It follows that if both homogeneous individual confidentiality and authenticity are satisfied then homogeneous individual security will be achieved, thus :-

$$Secu\ (N^2) \ \rightarrow\ Auth\ (N),\ Conf\ (N) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.14$$

4.2.4.2 Centralized security goals. There are three fundamental centralized security goals, namely :-

- *Conf (ta, 1)* which denotes centralized confidentiality. This is the channel to which the centre can read, but to which every party can only write.
- *Auth (ta, 1)* which denotes centralized authenticity. This is the channel to which the centre can write, but from which every party can only read.
- *Secu (ta, N)* which denotes centralized security. These are the N secure channels which the centre shares with every individual party.

Note that since *Secu (ta, N)* cannot be provided by *Conf (ta, 1)* and *Auth (ta, 1)* alone, two more centralized security goals are required, namely :-

- *Auth (ta, N)* which denotes authenticity of individual parties towards the centre. This the channel to which only a particular party can write, but from which only the centre can read.

- *Conf (ta, N)* which denotes confidentiality to the individual parties from the centre. This is the channel from which only a particular party can read, but to which only the centre can write.

Thus it follows that :-

$$Secu\ (ta, N) \ \rightarrow Auth\ (ta, N), Conf\ (ta, N) \qquad \dots\dots\dots\dots\dots\dots \ 4.15$$

4.2.5 Symmetric channels

Given that proof of authentication cannot necessarily be guaranteed by encryption it is generally advisable to use two separate algorithms and two corresponding cryptovariables to provide security in a symmetric channel. Access to the channel is then defined by knowledge of the particular cryptovariables. This leads to two definitions of security goals. Firstly *sym (secu, N^2)* is used to denote a secure channel between each pair of parties in a homogeneous symmetric system, where N^2 keys are needed (one for each channel). Secondly *sym (secu, N)* is used to denote a secure symmetric channel between every party and a centre in a centralized system, where N keys are needed (one for each channel).

The security mode *secu* suffices to define the security requirements of the above channel since the channel is symmetric and therefore there is no distinction between authenticity and confidentiality. This is because the knowledge of the relevant cryptographic key allows a party to both read from and to write to a particular channel. The use of a symmetric channel to achieve a given security goal requires the secure distribution of the necessary cryptographic keys. This means that to establish a secure channel between every pair of communicating parties using symmetric channels requires, by definition, another secure channel between every pair of communicating parties for the purposes of secure key distribution. This can be expressed as :-

$$Secu\ (N^2) \ \rightarrow sym\ (secu, N^2), Secu\ (N^2) \ \dots\dots\dots\dots\dots\dots\dots\dots \ 4.16$$

An interesting illustration of the properties in the use of a symmetric cryptosystem to achieve either confidentiality or authenticity, is that the security mechanism will *more* than satisfy the security goals and will also increase complexity, since N^2 keys need to be distributed securely for the N^2 symmetric channels. These properties are shown by the following logical productions :-

$$Conf\ (N) \rightarrow sym\ (secu, N^2), Secu\ (N^2) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.17$$

$$Auth\ (N) \rightarrow sym\ (secu, N^2), Secu\ (N^2) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.18$$

4.2.6 Asymmetric channels

Let *asym (mode, N)* denote a homogeneous cryptosystem where every communicating party has their own asymmetric channel. This requires that each of the *N* parties has to be able to distribute public keys between themselves and the other communicating parties. The security mode will be set to *auth*, *conf*, or *secu*. This results in :-

$$Conf\ (N) \rightarrow asym\ (conf, N), Auth\ (N) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.19$$
$$Auth\ (N) \rightarrow asym\ (auth, N), Auth\ (N) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.20$$

Here, the new security goal *Auth (N)* represents the problem of secure public key distribution between all communicating parties. The two security goals *conf* and *auth* may be combined to provide *secu* via two asymmetric channels; one providing authenticity and the other providing confidentiality. Here a sender of a message would first sign the message using a secret encryption key, and would then encrypt the result using the public encryption key of the receiving party. This gives rise to :-

$$Secu\ (N^2) \rightarrow asym\ (secu, N), Auth\ (N) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.21$$

Here *asym (secu, N)* represents both authenticity and confidentiality (via two asymmetric channels) for every communicating party. The new security goal is *Auth (N)*; the authentic distribution of public keys between communicating parties. Note that the complexity of the problem has now been reduced from the management of N^2 secure channels to the management of *N* authentic channels.

4.2.7 Trusted centres

Communicating parties may communicate via a trusted centre, rather than via direct communication with each other. This allows complete isolation between the communicating parties and can accommodate differing security modes and security mechanisms. The use of a trusted centre does, however, necessitate the centralization of security mechanisms rather than their distribution between the communicating parties concerned. This will affect the relevant layer of the security architecture. Let the following logical constructs be defined, as below :-

- *ta (secu, N)* which denotes the trusted authority of secrecy of party data.
- *ta (auth, N)* which denotes the trusted authority of authenticity of party data.
- *ta (conf, N)* which denotes the trusted authority of confidentiality of party data.

The case where a trusted centre shares a secure channel with every communicating party and acts as a communications hub can be described by :-

$$Secu\,(N^2) \rightarrow ta\,(secu, N), Secu\,(ta, N) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.22$$

If the security goal is only authenticity or confidentiality, then productions 4.23 and 4.24 apply respectively :-

$$Auth\,(N) \rightarrow Auth\,(ta, N), ta\,(auth, N), Auth\,(ta, 1) \qquad \ldots\ldots\ 4.23$$
$$Conf\,(N) \rightarrow Conf\,(ta, 1), ta\,(conf, N), Conf\,(ta, N) \qquad \ldots\ldots\ 4.24$$

If the security goal is either confidentiality towards the trusted centre or authenticity from the trusted centre, an asymmetric channel can be used, thus :-

$$Auth\,(ta, 1) \rightarrow asym\,(auth, 1), Auth\,(ta, 1) \qquad \ldots\ldots\ldots\ldots\ 4.25$$
$$Conf\,(1, ta) \rightarrow asym\,(conf, 1), Auth\,(ta, 1) \qquad \ldots\ldots\ldots\ldots\ 4.26$$

where *Auth (ta, 1)* is the authentic distribution of the public key of the trusted centre to all communicating parties.

4.3 Examples of Compound Logical Constructs

Typically, any complete cryptosystem will be a compound of more than one logical construct. Three examples are given below in Sections 4.3.1 to 4.3.3.

4.3.1 The Diffie-Hellman protocol for exponential key exchange

The Diffie-Hellman protocol [103] allows two communicating parties to agree on a common secret key whilst only communicating public information between them, without any prior exchange of secret information. The common secret key may then be used as a cryptovariable for a symmetric cryptosystem for subsequent secure communications between the two parties.

The essential aspect of the security of the system is to guarantee the authenticity of the publicly exchanged data. This gives rise to logical production 4.27, below :-

$$sym\ (secu,\ N^2),\ Secu\ (N^2)\ \rightarrow sym\ (secu,\ N^2),\ expka\ (N),\ Auth\ (N) \qquad \ldots\ldots\ldots\ldots\ 4.27$$

where *expka (N)* denotes the exponential key agreement.

There are *N* authentic channels required; one for each communicating party. Thus the complexity of the cryptosystem is *N*. Note that the cryptosystem cannot be implemented by the use of a symmetric channel. Note also that the exponential key exchange reduces the complexity from N^2 to *N*.

4.3.2 The certificate system for data authentication

The certificate system [28] is one which allows a trusted centre to authenticate the data of any individual communicating party. This is achieved by the party concerned sending the data to be authenticated (typically a public key) to the trusted centre, which then signs the data using its own secret key. The certified data is then made available to all communicating parties, thus :-

$$Auth\ (N)\ \rightarrow phys\ (auth,\ N),\ ta\ (auth,\ N),\ asym\ (auth,\ 1),\ Auth\ (ta,\ 1) \qquad \ldots\ldots\ldots\ldots\ 4.28$$

Thus a cryptosystem with *N* authentic public keys can be implemented using *N* authentic physical channels; one from each communicating party to the trusted centre *ta*, which distributes the public keys using an asymmetric channel for the purposes of authenticity. Such authenticity of the public keys of the communicating parties can only be achieved if the public key of the *ta* is distributed authentically. Note that no secret information of the communicating parties is entrusted to the trusted centre and also that the complexity of the security management is reduced from *N* to 1 by the use of a trusted centre.

4.3.3 The identity-based asymmetric cryptosystem

This idea, originally proposed by Shamir [80], proposes that communicating parties may use their individual identities as their public keys, allowing either read or write access (but not both) to their asymmetric channels. To allow any communicating party to receive an authentic copy of the public key of any other communicating party, the services of a trusted centre are required. The three-part procedure for registering an authentic channel with the trusted centre is as follows :-

- Each communicating party must first physically register its identity with the trusted centre *ta*. This constitutes an authentic physical channel between each individual communicating party and the trusted centre.

- The *ta* then computes the secret key of the individual communicating party from its identity, using the secret key of the trusted centre. The resultant key is returned to the individual communicating party by means of a tamper-proof device which provides both physical and logical security. This creates a secure physical channel between the trusted centre and each individual communicating party.
- Each individual communicating party is now able to sign its messages and all other receiving parties may verify that signature using the public key of the originating party.

Thus an identity-based cryptosystem may be modelled by the following logical construction :-

$$Auth\ (N)\ \rightarrow\ phys\ (auth,\ N),\ ta\ (secu,\ N),\ phys\ (secu,\ N),\ (Auth\ (N)) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ 4.29$$

Thus the security goal of establishing asymmetric channels for each communicating party may be achieved by the use of *N* authentic physical channels, a trusted centre for computing and securely storing the secret keys of the individual communicating parties, and *N* secure physical channels for the subsequent distribution of the secret keys. Note that the it is intended that the essential advantage of such an identity-based system is that the identities (and hence the public keys) of each individual communicating party can be globally distributed between all other communicating parties.

4.4 Systematic Design and Analysis of Cryptographic Protocols

There are many relevant works on the subject regarding the systematic design of cryptographic protocols and of their analysis. There are several formal methodologies for reasoning about security and several techniques have been developed to practically apply such methodologies, even if in only a limited manner, as discussed with regard to BAN-type logic in Section 4.1.2, and with regard to alternative logics in Section 2 above.

4.4.1 Classification of formal description and design methodologies

The work of Knapskog [104] has identified three formal description and design methodologies, namely :-

- Abstract Syntax Notation (ASN). This is specified in a joint CCITT / ISO standard [105]. It is a semi-formal tool for defining protocols in the form of high level languages. The tool does not necessarily preclude incomplete or ambiguous specifications.

- Functional Specification and Description Language (SDL). This is a development of the extended Cryptographic Finite State Machine (CFSM) modelling technique [27]. It allows generation of code for a software/firmware/hardware implementation of the graphic or textual representations of the model.

- Numerical Petri Nets (NPN). This method has been used extensively for general protocol description and construction. Varadharajan [106], has shown how a Petri net formalism may be used to model information flow security requirements. The proposed framework can be use to specify a range of information flow security policies and it is asserted that the approach is particularly suitable for modelling security aspects in distributed systems. Tools for verification purposes have also been developed.

The above methods may be applied to the (incorporation of) security mechanisms in the various layers of OSI architecture. Security mechanisms may be incorporated within the following ISO layers :-

- Physical layer.
- Network layer.
- Transport layer.
- Presentation layer.

The majority of the development regarding the incorporation of security mechanisms into communications protocols has been in the physical and the network layers; where node-to-node encryption and decryption functions are appropriate.

4.4.2 The systematic design of nonce-based protocols for authentication by cryptographic challenge and response

There are many different proposals for authentication protocols which are based on challenge and response. Such protocols may be fundamentally two-party or three-party. Two-party authentication protocols allow two parties to perform either a one-way or a two-way authentication exchange, which may then be expanded to incorporate the authentication of successive other parties. Three-party authentication protocols provide the same security functionality as above, but require the use of a trusted intermediary: a trusted authentication centre or host. Refer to Yahalom *et al* [107].

The properties of two-party authentication protocols are of particular interest in application to personal communication systems, since such systems generally involve the authentication of only two parties; the mobile subscriber and the host network operator. It is thus both unnecessary and undesirable, with respect to cost and signalling/processing efficiency, to incorporate a third party purely for the purposes of an authentication host.

An additional consideration in the design of such authentication protocols is the type of perturbation which is to be incorporated into values such as challenge vectors and cryptovariables. With a view to challenge-response authentication protocols, the challenge vectors must be selected with extreme care to minimize the probability of replay and cryptanalysis attacks. To this end they are generally either time-stamped (with a time-related data element or with a serial number) or they are selected on a (pseudo)random basis from a large set of possible values. The latter type of value is known as a nonce, and is particularly effective in authentication schemes by providing high perturbation and not requiring any locally-kept time reference.

A significant advance in the theory of authentication protocol security has been made by Bird *et al* [108], who demonstrate that many cryptographic challenge-response protocols can be broken by an analysis of the classes of data exchanged between authenticating parties, to such effect that an attacker may actively initiate several authentication sessions with a view to impersonating one of the parties by recovering an appropriate authentication response from an exchange with another party. Their technique for establishing whether such protocols may be broken is based on the compilation of a list of values an attacker would have to send, in comparison to a list of values a *bona fide* authenticating party could send during such an authentication exchange. If the latter list contains the required values of the former list, even if recoverable by easy computation, then the protocol will be breakable by using their attack technique. An example of an authentication protocol which is susceptible to such attack is given in Figure 4.1 below :-

A ***B***

$E\,(A + E\,(N_1))$

⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒

$N_1,\, E\,(B + E\,(N_2))$

⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐⇐

N_2

⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒⇒

Figure 4.1 A bi-directional two-party authentication exchange

The Figures depict information exchange between two parties, *A* and *B*. *N* represents a nonce and *E*(*N*) represents the nonce enciphered by encryption function *E*.

The above authentication exchange may be successfully broken using the above attack technique, where a parallel authentication session is initiated by the attacker in order to receive the authentication vector required. Such an attack, involving an interleave of two parallel authentication sessions, is known as an interleaving attack. Such an attack on the above authentication protocol is shown in Figure 4.2 below :-

$$\begin{tikzcd}
\boldsymbol{A} & \boldsymbol{X} & \boldsymbol{B} \\
{} & {} \arrow[r, Rightarrow, "E(A + E(N_1))"] & {} \\
{} & {} & {} \arrow[l, Rightarrow, "{N_1, E(B + E(N_2))}"'] \\
{} & {} \arrow[l, Rightarrow, "E(B + E(N_2))"'] & {} \\
{} \arrow[r, Rightarrow, "{N_2, E(B + E(N_3))}"] & {} & {} \\
{} & {} \arrow[r, Rightarrow, "N_2"] & {}
\end{tikzcd}$$

Figure 4.2 Parallel session (interleaving) attack on the authentication protocol (as defined in Figure 4.1)

This attack can be thwarted by utilizing an identity-based protocol or, preferably, a protocol which incorporates a nonce-based value in every response to a challenge vector. This will prevent an attacker computing an appropriate response to a challenge vector by referring to the responses to previous challenges.

Bird *et al* postulate that there are four specific security weaknesses which must be avoided if any two-party challenge-response authentication protocol is to be provably secure, namely :-

- The protocol should add some (random) perturbation to the selected challenge vector, both in order to prevent any combination of parallel sessions by an attacker for the purposes of reference and in order to prevent chosen-plaintext attacks.

- The identity of authenticating parties should be incorporated into their responses to challenge vectors in order to guarantee that the responses to a given challenge vector are different for all parties. This renders unfeasible the computation by an attacker of the responses of one party from the responses of another party and hence prevents any parallel session attack.

- Every individual data flow should be independent of any other data flow within the protocol to such effect that it is unfeasible for an attacker to compute any data flow within the protocol from other data flows. This is in order to prevent an interleaving attack.

- Data fields within vectors should be cryptographically separated such that an attacker cannot control one field by manipulating another field. This supports the security function of each data field and may be achieved by authenticating a concatenation of all the data fields. Such authentication may be achieved by means of a cryptographic check sum such as a message authentication code, or MAC.

The authors propose a secure two-party challenge-response authentication protocol which incorporates the above security design considerations is shown in Figure 4.3 below :-

$$\begin{array}{lcr}
A & & B \\
 & N_1 & \\
 & \Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow & \\
 & E\,(B + E\,(N_2 + E\,(N_1))),\, N_2 & \\
 & \Longleftarrow\Longleftarrow\Longleftarrow\Longleftarrow\Longleftarrow\Longleftarrow & \\
 & E\,(N_2 + E\,(N_1)) & \\
 & \Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow\Longrightarrow &
\end{array}$$

Figure 4.3 A secure two-party challenge-response authentication protocol

Their proposed solution depends largely upon the inherent security of the CBC mode of the DEA. This gives rise to cryptographic products based on the following general equation :-

$$C = E\,(a + E\,(b + E\,(c))) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 4.30$$

where E	=	encryption transformation
C	=	ciphertext
a, b, c	=	constants, random values and modulo-2 sums of these

It can be shown that the security of the above protocol reduces to that of the DEA in CBC mode, thus providing a provable level of security; this depending upon the actual level of security provided by the DEA.

4.5 Conclusions

This chapter has presented an overview of various formal logical methodologies which may be applied to the specification, implementation and verification analysis of cryptographic key distribution and authentication protocols.

It has been shown how a formal language may be used to describe security goals and mechanisms. It has also been shown that such techniques allow security architecture completeness to be proven when the security goals are satisfied. It has been further shown that the management of complexity may be aided by the introduction of a complexity factor into each security goal and security mechanism, allowing such methodologies to be used both as design tools and as evaluation tools.

Work has shown that authentication protocols which use smart cards offer a significant security improvement over those which use simple passwords. Similarly, the use of formalisms has aided the comparison of the trust relationships on which the above protocols are based. The fact that every authentication protocol has individual characteristics which may allow it to be used to counter specific security threats, whilst possessing its individual security limitations and resulting in specific technological requirements, has been considered with regard to various logics for reasoning about security.

It has been shown how the BAN approach to protocol security evaluation requires further development to allow for the detection of non-termination of protocols, but that it is unlikely that there is an extension to BAN suitable for the analysis of zero-knowledge protocols.

It has been shown how the rôles of the principals of cryptographic key distribution protocols must be clearly defined before certain types of formal analysis technique, such as BAN, will identify all possible security flaws in all examples of the appropriate types of key distribution protocol.

The use of Abstract Syntax Notation (ASN) as a semi-formal tool for defining protocols in the form of high level languages has been presented. It has been shown that the tool does not necessarily preclude incomplete or ambiguous specifications.

The concept of the functional Specification and Description Language (SDL) has been presented as a development of the extended Cryptographic Finite State Machine (CFSM) modelling technique, which allows generation of code for a software / firmware / hardware implementation of the graphic or textual representations of the model.

It has been suggested that the best solution for a formal logical analysis tool will be a combination of the ASN and SDL methodologies. The use of numerical Petri nets has been identified as a potentially valuable tool for the purposes of protocol description and construction.

The full benefit of the ISDN will only be achievable if security mechanisms are incorporated within the application layer and below. This will require very sophisticated techniques for formal description, construction and verification.

CHAPTER 5. ANALYSIS OF THE SECURITY PROVISIONS OF SECOND-GENERATION PERSONAL COMMUNICATION SYSTEMS.

This chapter presents an analysis of the authentication and ciphering security provisions of the Second-Generation PLMN systems; GSM900/DCS1800 and DECT. The security level afforded by the cryptographic algorithms employed in these systems is considered in relation to the initial probability of successful cryptanalysis. The system security architecture is evaluated using a formal methodology, as developed in Chapter Four. The formal methodology is used to analyse the security goals and the security mechanisms of the PLMNs concerned.

The various security mechanisms provided, by both the GSM900/DCS1800 PLMN and the DECT PLMN, are compared with regard to the security requirements of mobile subscribers and network service providers, as outlined in Chapter One.

5.1 Introduction

The advent of the Second-Generation PLMN systems has enabled cryptosystems to be readily incorporated by virtue of the intrinsically digital nature of the Second-Generation systems. In addition to supporting all the communications functionality associated with mobility in such networks, security functionality has to be simultaneously accommodated. Such security functionality must be implemented by means of associated cryptographic algorithms and protocols.

The purpose of this chapter is to present an analysis of the (maximum) security level afforded by Second-Generation PLMN systems, in addition to a formal analysis of the completeness of their associated protocols. The two Second-Generation systems chosen for this analysis are the ETSI GSM900 cellular telephone system (and its PCN variant, DCS1800) and the ETSI DECT cordless telephone system.

The threat of attack to these systems will be considered and certain attack scenarios will be presented.

5.2 Overview of the ETSI GSM900/DCS1800 PLMN system security provisions

5.2.1 Categories of security

The security measures taken in any of the various digital cordless or cellular telephone systems fall into four broad categories. These are outlined below, followed by an analysis of how these categories are satisfied within the GSM900/DCS1800 and DECT communication systems.

5.2.1.1 Protection of User Data. Since any data, be it speech or alphanumeric, which is transmitted over a radio telephone system is susceptible to eavesdropping when relayed over the air interface, measures have to be taken to counteract any such attack of the system.

The only realistic means of guaranteeing such communications security is to employ traffic encryption. This requires that there be a suitable cryptographic algorithm implemented at both the base station controller and the mobile station, to enable bulk traffic encryption. Any cryptosystem necessarily requires suitable schemes for cryptosynchronization and cryptographic key management, refer to Cooke and Brewster [109]. The cryptographic protection of traffic will enhance any existing eavesdropping security, which may be due to the digital nature of the traffic, any frequency hopping and TDMA operation.

5.2.1.2 Protection of Signalling Information. Any data which may accompany the user data to identify subscribers or their locations must be protected such that there is no possibility of an eavesdropper discovering this information. This is to comply with the user requirement for party anonymity and location confidentiality. Such protection may be implemented by means of ciphering of transmitted data or by the use of pseudonyms. The latter is used in the GSM900/DCS1800 PLMN, refer to Section 5.2.4.4.

5.2.1.3 Fraud Prevention. In any credit-based telecommunication system there has to be a highly reliable means of ensuring that all calls connected (and other services provided) are billed to the appropriate party, and that no calls or services are not billed or are billed to the wrong party. Thus any system for billing calls or services must be tamper-proof. Such protection may be implemented by virtue of party authentication and the encryption of signalling information.

5.2.1.4 Equipment Verification and Entity Authentication. It is essential that in any public radio telephone system there is an effective means of verification of subscriber equipment and also of authentication of network entities, both mobile and network. It must be ensured that any subscriber equipment operating with the system conforms to stringent specifications, is operating properly and has not been "black listed". It is equally important that the identity of any party using the network is positively authenticated, to ensure correct billing, authorised call connection and provision of attributed facilities. There may also be a requirement for the authentication of the service provider by the mobile subscriber. Equipment verification will be based upon a scheme incorporating an electronic serial number, entity authentication will invariably be based upon a cryptographic authentication exchange.

5.2.2 Encryption within the GSM900/DCS1800 coding scheme

The arrangement adopted for the GSM900/DCS1800 PLMN is typical for mobile radio systems, in that speech coding is first applied to compress the base-band data rate. Refer to Figure 5.1, below, for an illustration of the adopted coding arrangement for incorporating encryption in the GSM system. The resultant data may be subsequently mixed with other data signals and will be mixed with synchronization data. The net bit rate for speech transmission is reduced to 13.0 kbit/s; bit rates for data services being in the range of 75 bit/s to 9.6 kbit/s. In either speech or data transmission modes the gross bit rate is 22.8 kbit/s due to the addition of error protection by channel coding. Note that the degree of error protection differs between the speech and data modes. Further details are given in Vary [110].

Bit interleaving is applied to the channel coded bit stream to afford a degree of robustness against the burst errors typically encountered in the mobile channel. The bit interleaving is followed by channel encryption to provide a level of privacy to both the user data and the signalling information. The base station then combines 8 subscriber channels by time-domain multiplexing. Global synchronization information is then added, followed by GMSK modulation of the final 270.8 kbit/s multiplexed signal. Note that frequency hopping may be optionally applied to combat frequency-selective interference and fading, by changing the instantaneous carrier frequency between time slots.

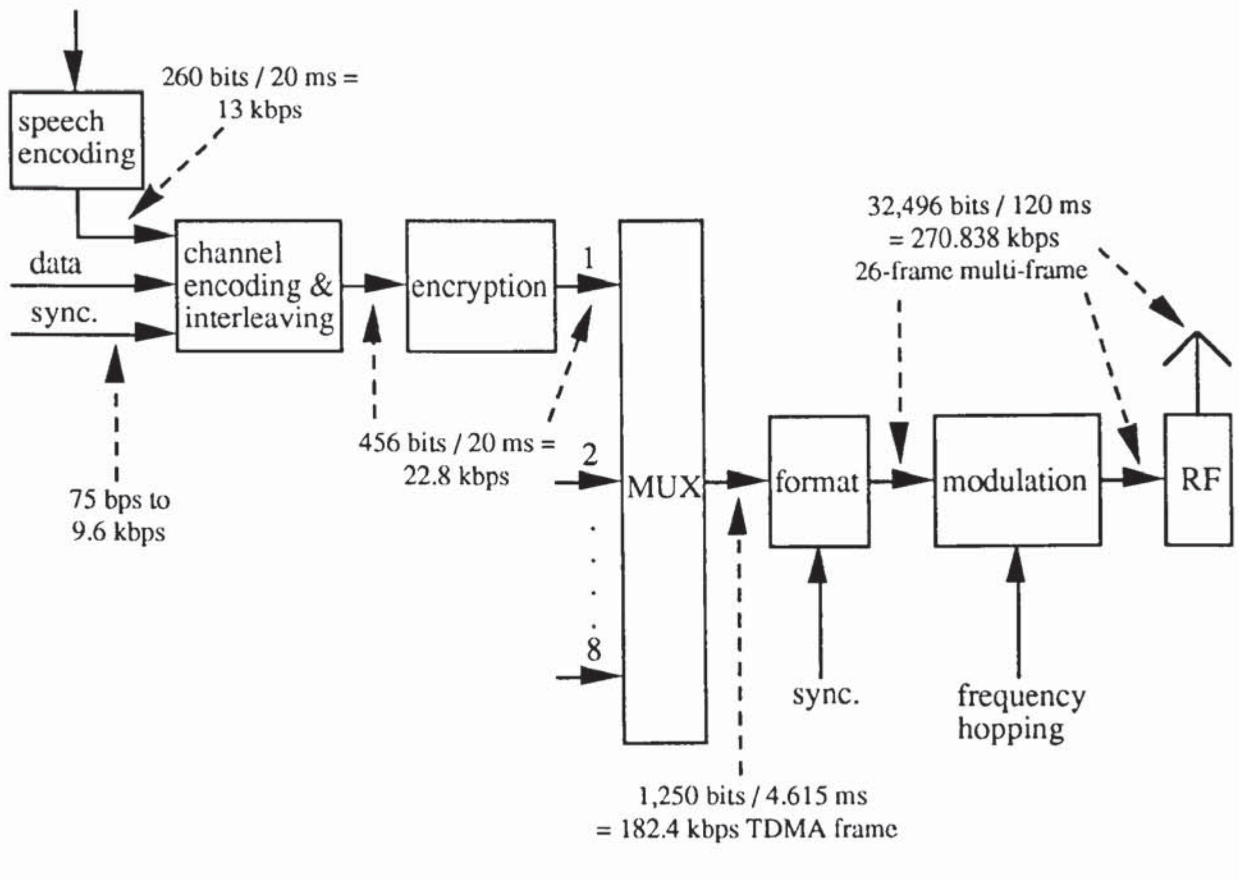
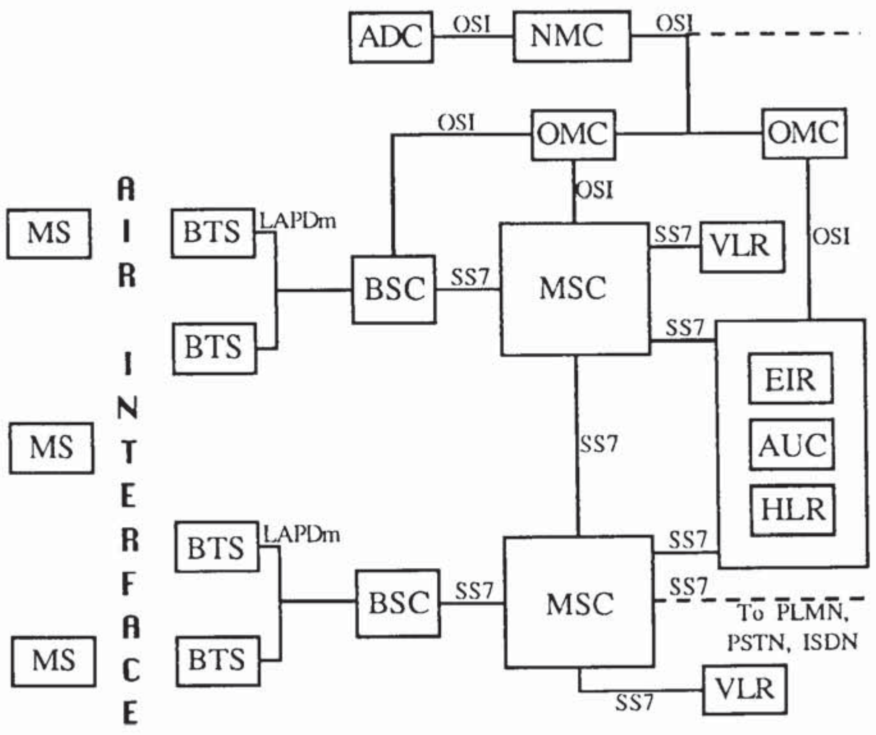


Figure 5.1 Transmitter Part of Base Station

5.2.3 The GSM900/DCS1800 PLMN intelligent network

The intelligent network architecture of the GSM900 system (and its PCN variant, DCS1800) is illustrated in Figure 5.2 below. This illustrates the connectivity between the various network elements; data bases, switches and transceivers. The functionality of the various network elements crucial to the implementation of communications security is explained below.



ADC:	Administration Centre	AUC:	Authentication Centre	BSC:	Base Station Controller
BTS:	Base Transceiver Station	EIR:	Equipment Identity Register	HLR:	Home Location Register
MS:	Mobile Station	MSC:	Mobile Switching Centre	NMC:	Network Management Centre
OMC:	Operation and Maintenance Centre			VLR:	Visitor Location Register

Figure 5.2 GSM900/DCS1800 PLMN Network Architecture

5.2.4 GSM900/DCS1800 PLMN Security Implementation.

The following overview of the GSM system security features should be read with reference to Figures 5.3 to 5.5, which illustrate the network architecture relevant to the security functionality. Figures 5.6 to 5.8 are protocol diagrams associated with the security functionality. There are three data bases fundamental to the operation of the authentication and encryption functions of the GSM system, refer to Cooke and Brewster [111]. These are explained below.

5.2.4.1 Authentication Centre (AUC). This data base contains the subscriber's authentication parameters as data pairs; each pair contains a subscriber's IMSI

(International Mobile Subscriber Identity) and K_i (the subscriber's corresponding secret cryptographic key). The IMSI / K_i pairs are stored in encrypted format.

There is encrypted communications with the ADC (Administration Centre) from where the various subscriber identities and attributes *etc* are controlled. The algorithms within the AUC generate security-related parameters. These algorithms depend upon the particular Public Telecommunications Operator (PTO) concerned.

5.2.4.2 Home Location Register (HLR). This is a secure data base containing a permanent record of subscription parameters and current subscriber locations for the particular Public Land Mobile Network. All data is referenced to the individual IMSIs.

5.2.4.3 Visitor Location Register (VLR). This is a secure data base similar to the HLR, except that it contains all relevant parameters of mobile subscribers currently within the location area of this particular VLR.

There are two fundamental aspects of the security of operation in the GSM system; namely the authentication of communicating parties and encryption of the communications traffic flowing between them.

During the initial stage of call set-up, every mobile subscriber has to undergo an authentication procedure, followed by encryption of all traffic, refer to Cooke and Brewster [112]. The procedure is detailed in Section 5.2.4.4. and is specified in the ETSI GSM recommendations [113].

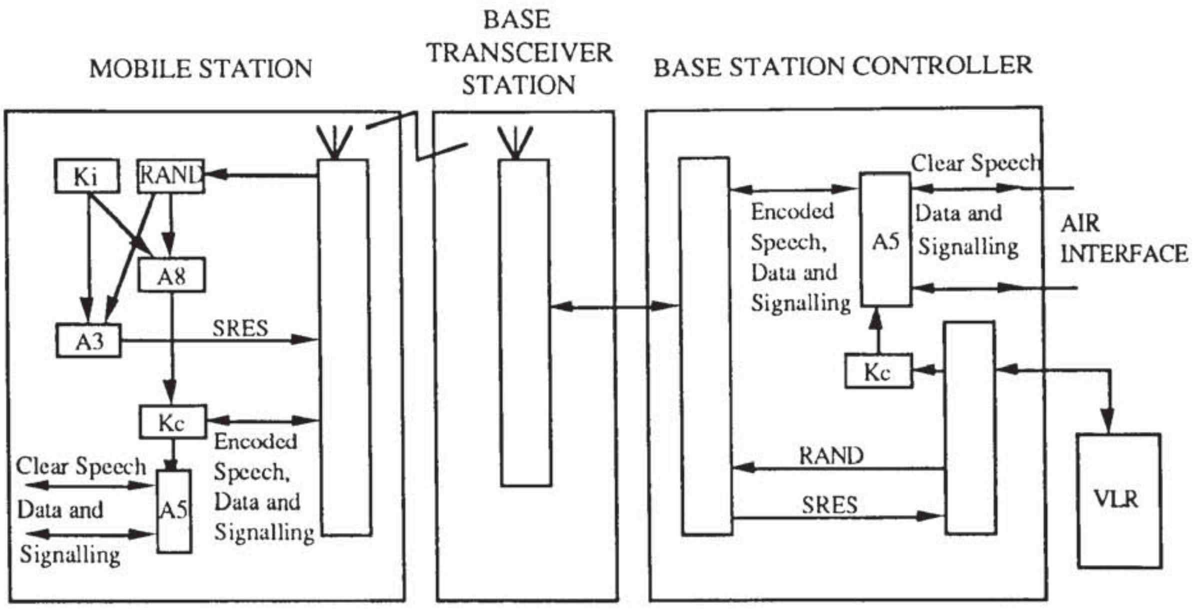


Figure 5.3 Authentication and Ciphering Procedure for the GSM900/DCS1800 PLMN

5.2.4.4 Authentication and Encryption Procedure. At the start of each new call between the mobile station and the network, the mobile station is required to authenticate itself to ensure that the identity of the mobile station is proven. This is followed by bulk encryption of all traffic (speech, data and signalling information) transmitted between the network and the mobile station.

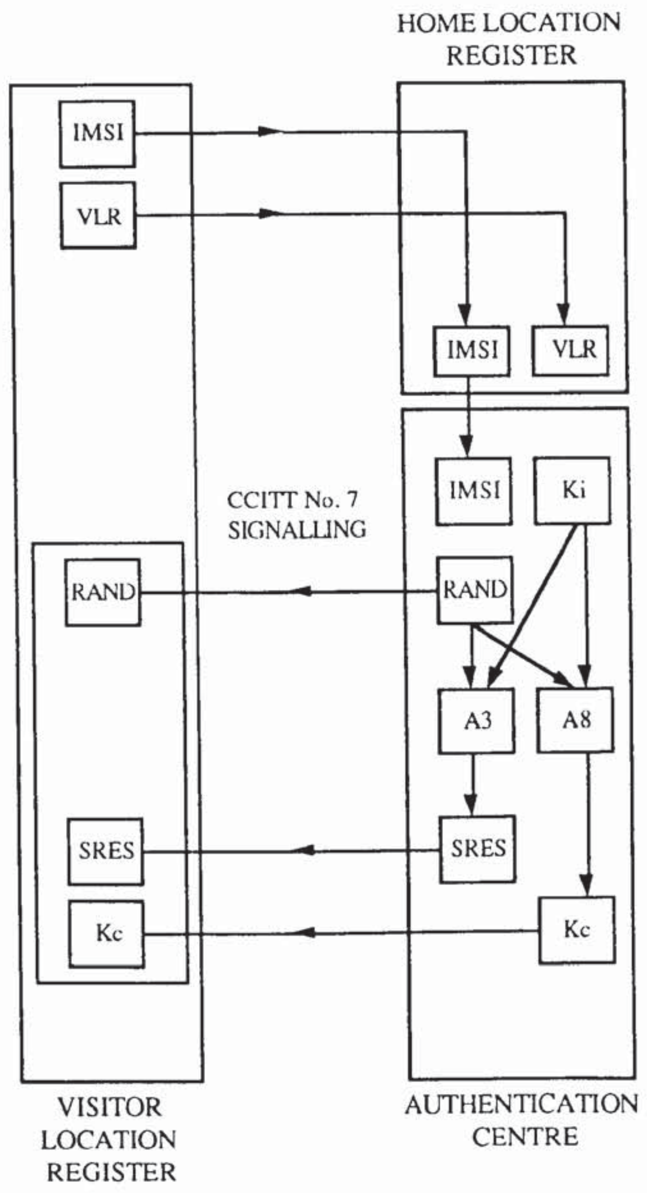


Figure 5.4 Network Connectivity for Authentication and Ciphering in the GSM900/DCS1800 PLMN

When a request is received by the network from a mobile station for a call to be connected, the mobile station identity is received in the form of a Temporary Mobile Subscriber Identity (TMSI), refer to Cooke and Brewster [112], which would have been forwarded to the current VLR from the last VLR. Once a TMSI has been used a new TMSI is calculated by the

VLR and sent in encrypted form to the mobile station, refer to ETSI GSM recommendations [114]. When the VLR receives a TMSI from a mobile station during authentication, it is used to look up the corresponding value of International Mobile Subscriber Identity (IMSI), which would also have been forwarded by the last VLR. The value of IMSI is sent to the HLR, where the process detailed below occurs, refer to Audestad [115]. Refer to Figure 5.6 for the relevant protocol diagram of the procedure for outgoing call set-up. The circles represent the relevant GSM900 procedure calls.

The value of IMSI is sent to the AUC, which stores pairs of IMSI and the corresponding value of K_i (the individual mobile subscriber's secret cryptographic key) in a secure data base. The value of K_i is used in conjunction with a random number RAND to calculate a signed response SRES using an algorithm A3. The same values of RAND and K_i are used to calculate a traffic encryption key K_c using an algorithm A8. The three variables form an unique authentication set {RAND, SRES, K_c}, which is different for every call made. For this reason the AUC calculates several authentication sets at a time, these being forwarded to the relevant HLR. One authentication set is forwarded from the HLR to the VLR currently controlling the mobile subscriber, as and when requested.

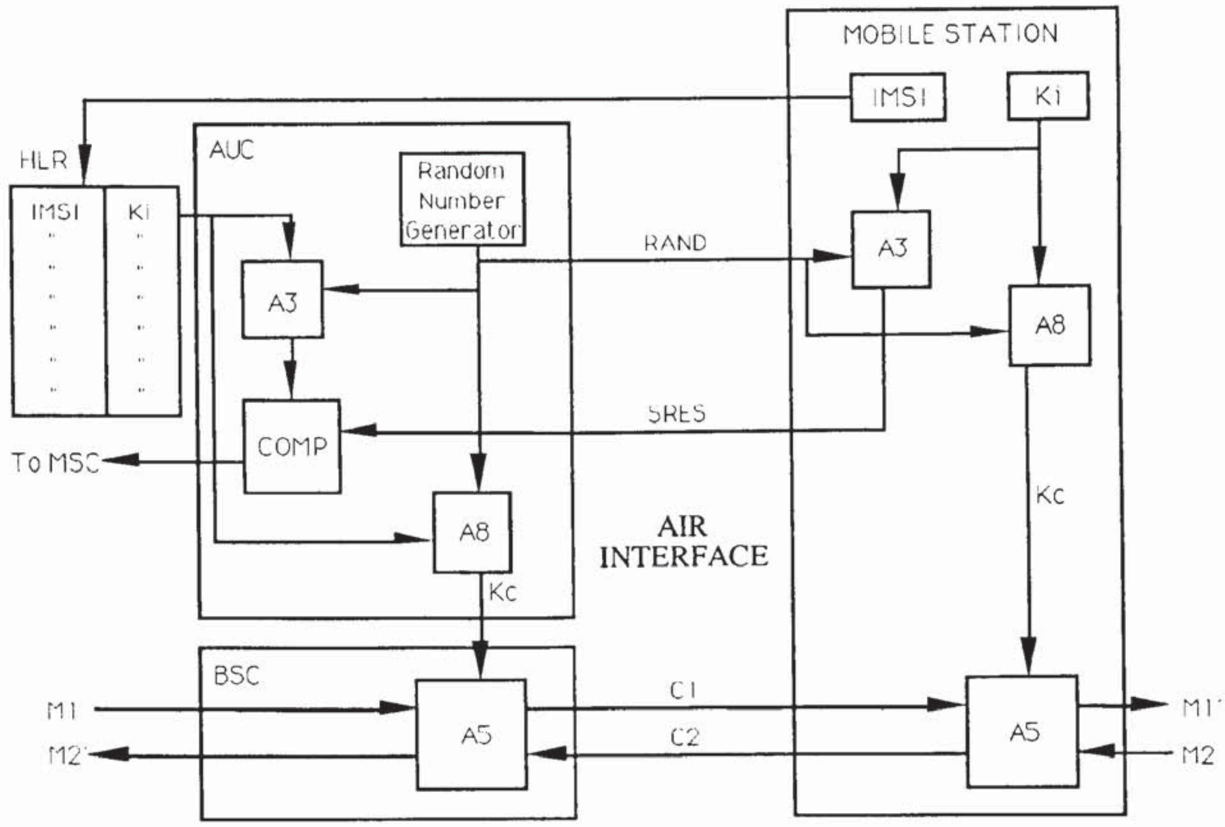


Figure 5.5 Authentication and Ciphering Procedure for the GSM900/DCS1800 PLMN

The VLR then forwards the value of RAND to the mobile station via the Base Station Controller. The mobile station performs the same calculations of SRES and K_C using algorithms A3 and A8; it has a copy of K_i held securely in the subscribers smart card. The value of SRES calculated by the mobile station is transmitted back to the Base Station Controller where it is forwarded to the VLR. It is checked to see if it matches the value of SRES already held by the VLR.

If the values of both SRES are identical the identity of the mobile subscriber is considered to have been positively authenticated, whereupon the MSC allows the mobile station to be call connected.

Since the Base Station Controller and the mobile station both now possess the value of K_C, they are able to communicate securely by encrypting all communications between them using K_C as key input to a third algorithm A5. This symmetric algorithm performs a bulk encryption / decryption on all communications traffic.

Similarly the relevant protocol diagram of the procedure for incoming call set-up is shown below in Figure 5.7.

5.2.4.5 Location updating and downloading of data. The relevant procedure set to perform location updating is shown below in Figure 5.8. The location update process is initiated by the MS, based on identity information particular the the relevant cell, which is broadcast via a control channel. Note that the MSC is transparent during this process. The process is supervised from the VLR by procedure P1. Other procedures may be invoked, if required, namely :-

- (From the previous VLR). To "send parameters" requesting the previous VLR to provide IMSI and authentication parameters.
- (With the MS via the MSC). To "provide IMSI" (if IMSI is not obtained from the previous VLR), to "authenticate" (if the MS is to be authenticated), to "set cipher mode" (if a new TMSI is allocated to the MS) and to "forward new TMSI" (for passing the new TMSI to the MS in ciphered mode).
- (With the HLR). To "send parameters" requesting the HLR to return authentication parameters and to "update location" to send location information to the HLR. The location information is used by the HLR to route the calls to the MS.

The downloading of subscriber parameters to the VLR is performed by means of a separate procedure, "update category and supplementary services", which enables the VLR to handle the majority of calls to and from the MS without interrogating the HLR.

The entry for the MS may now be deleted from the previous VLR. The "cancel location" procedure is controlled by the HLR.

Thus it can be seen that location updating achieves an update of the HLR for routing of incoming calls to the MS and also achieves an update of the VLR with the relevant subscriber parameters for autonomous service handling by the VLR.

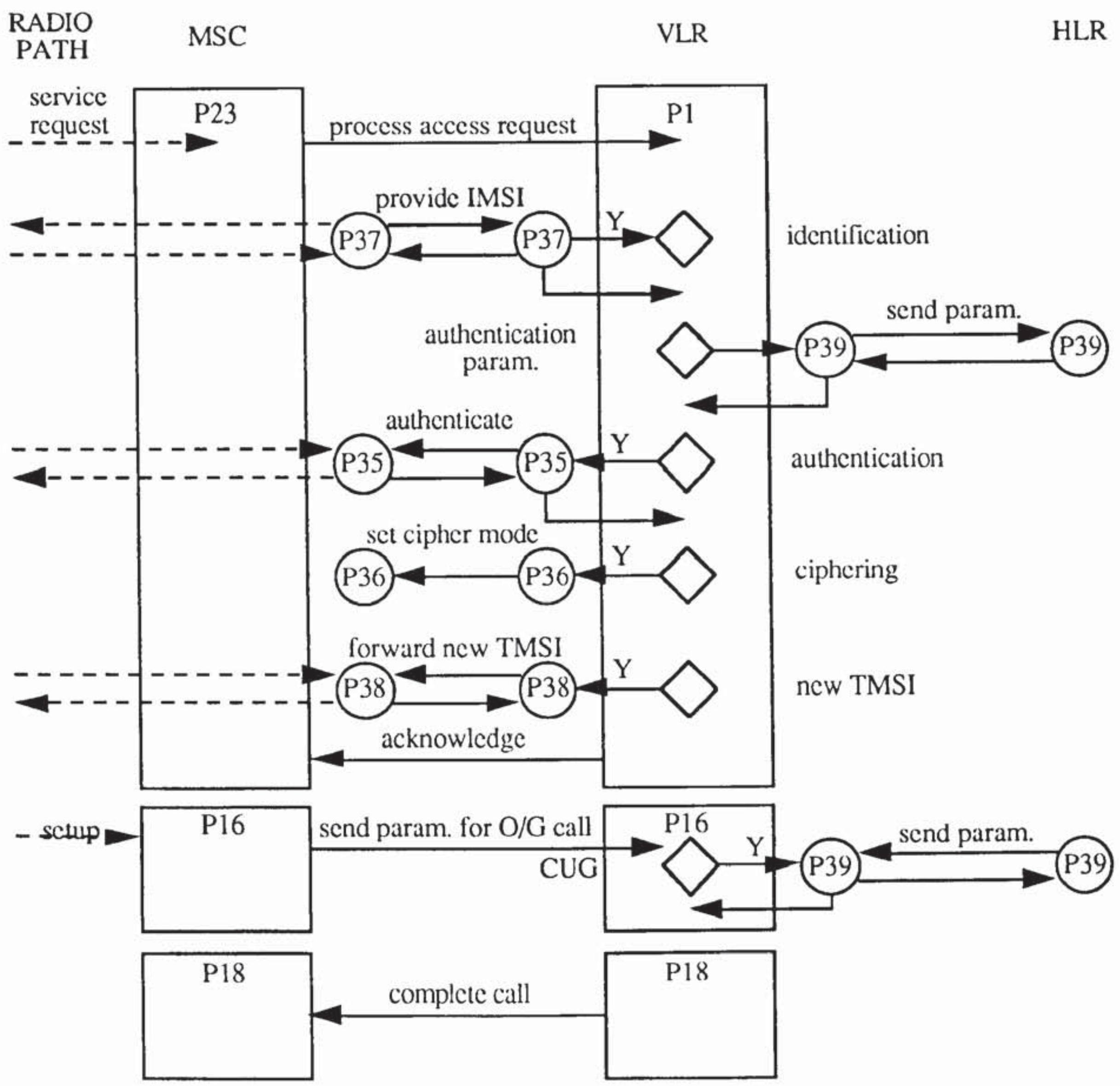


Figure 5.6 Procedure for Outgoing Call Set-up

5.2.5 GSM900/DCS1800 PLMN control channels.

The Common Control CHannel (CCCH) is a combination of common control channels. They are used during the establishment of a connection between a MS and a BS before a dedicated control channel has been allocated. There are three downlink-only channels; one for MS paging, one for access grant and one for cell broadcast. There is one uplink-only channel for random access attempts, refer to ETSI GSM recommendations [116].

There are three basic types of dedicated control channel, namely Stand-alone Dedicated Control CHannels (SDCCH), Slow Associated Control CHannels (SACCH) and Fast associated Control CHannels (FACCH).

The SDCCH carry signalling information following MS-BS connection establishment and channel assignment.

The SACCH is always mapped onto the same physical channel as a traffic channel or a SDCCH. It carries general information between MS and BS such as the measurement results from the MS about current and neighbouring cell signal strengths.

The FACCH carries the same signalling information as the SDCCH, but is only assigned when a SDCCH has not been assigned and obtains access to the physical resource by stealing frames from the traffic channel to which it has been assigned.

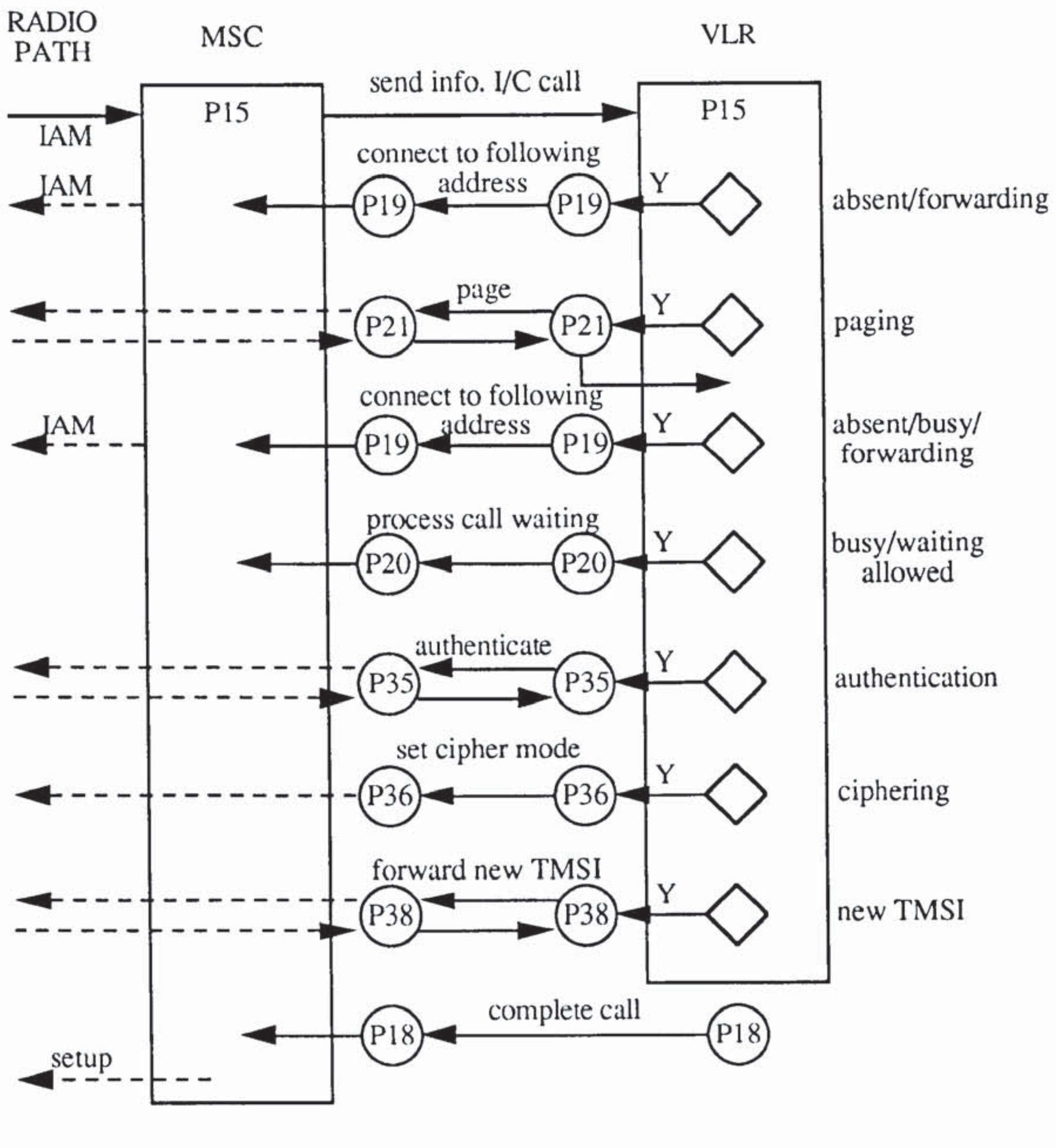


Figure 5.7 Procedure for Incoming Call Set-up

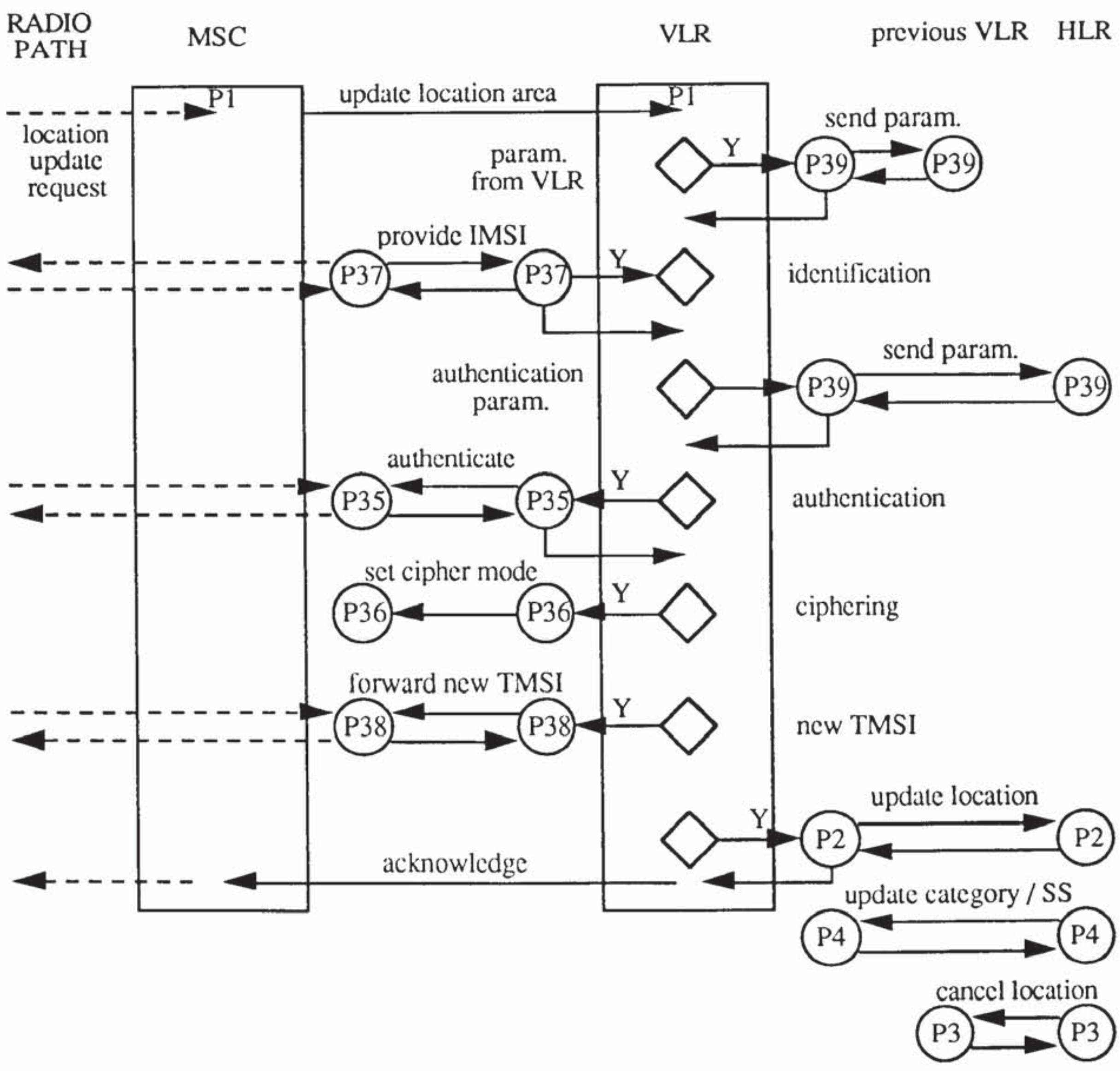


Figure 5.8 Overall Procedure for Location Updating

5.2.6 GSM900/DCS1800 PLMN burst types.

The GSM burst types are shown below in Figure 5.9. The normal burst is that transmitted between MS and BS during a call. There are 116 bits of encrypted data; the data only being encrypted if the encryption process has ben initiated by the network. The remaining bits are start bits, stop bits and a training period.

The frequency correction burst has the same bit structure as the normal burst, except that the encrypted bits and the training sequence are replaced by a 142-bit fixed series of zeros. This burst enables frequency synchronization of the mobile station and also enables the mobile station to readily find the broadcast channel.

The synchronization burst is used for time synchronization of the mobile. It has the same bit structure as the normal burst except that the encrypted bits are partly replaced by an extended training sequence of 64 bits.

The dummy burst has the same bit structure as the normal burst except that it carries no data. The encrypted bits are replaced by mixed bits; these being a known sequence of bits with a mean value of a half. This burst is used in place of a normal burst when no other data is being transmitted.

The access burst is used by the mobile station to access the system. It has an extended guard period of duration 68.25 bits to accommodate burst transmission from a mobile station when the timing advance is not known at the first access.

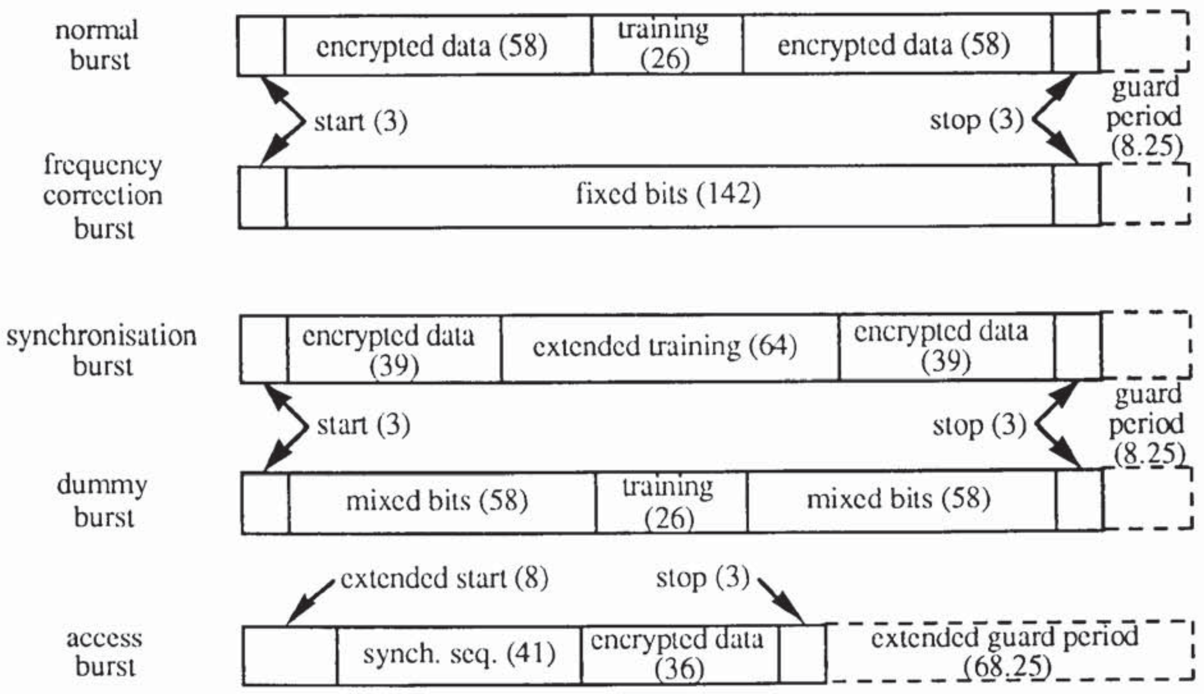


Figure 5.9 Burst Types in the GSM PLMN

Normally a MS identifies itself within a given location area by sending out a TMSI. Outside the location area the location area identification (LAI) must also be sent. If the LAI and the TMSI do not match the MS is requested by the network to send its IMSI. A new TMSI is subsequently sent to the MS in encrypted form, refer to Hodges [117].

The same ciphering key, K_C, is used for both directions of traffic, since A5 is a symmetric algorithm. The ciphertext output of algorithm A5 is generated from its plaintext input (output from speech codec) and is perturbated by K_C and the TDMA frame number. The ciphertext is in the form of 114-bit words. Two stealing flags are added and the final 116 normal burst bits are produced.

Ciphering is initiated in the MS when it receives a "start cipher" message from the BS. The BS initiates its ciphering process as soon as it receives valid ciphered data from the MS. Cryptosynchronization is achieved by means of the TDMA frame numbers. Note that K_C will be updated upon every cell hand over, upon every new call connection and whenever the network operator requests it, however the the change of K_C at cell hand over does not take the form of a full ciphering key update; the new key is created from the old key by changing some of the bits in a deterministic manner.

The identification of the MS by its IMSI has to be permitted on the basis that the relationship between IMSI and TMSI may become lost and also that it may not be immediately possible to use a common ciphering key for the purposes of allocating a TMSI. The latter situation may occur because the key used for encrypting a new TMSI is a by-product of a preceding authentication and such authentication does not always follow identification.

Note that several authentication triplets may be stored for use by the MSC, so that if the link to the HLR by the AUC concerned is lost, then authentication may continue. This has the additional advantage of not distributing security sensitive subscriber data over an (international) network.

5.3 Evaluation of GSM900/DCS1800 PLMN System Security Provisions

5.3.1 The threat of attack.

In order to quantitatively evaluate the security level of any communications system, it is necessary to have a reference level of security which is determined from an assumed level of threat of attack to that system. It is generally accepted that the security level offered by any public cordless or cellular telephone system should be at least that provided by the PSTN. This applies to all categories of security, namely those concerning user data, signalling information, equipment verification / subscriber authentication and fraud prevention.

However, the nature of the threat of attack to any mobile system is necessarily greater than that to the PSTN, since communications between the network and the mobile station are via the air interface, resulting in their ready availability for passive or active attack within the radius of a cell (which may be up to several Kilometres).

Although it is true that the various digital cordless and cellular telephone systems have intrinsic properties which make casual eavesdropping upon their communications harder than it might otherwise be (*eg*. the digital nature of the data, TDMA techniques and frequency hopping), a dedicated attack upon the system may still prove to be possible. Such an attack might require relatively sophisticated signal processing and cryptanalysis, but it cannot be entirely ruled out.

It is for the above reason that the assumed level of the threat of attack upon the system must be carefully considered in conjunction with the nature of the data to be protected and the requirements of the users of such a system. This will lead to a desired level of security resulting from the inclusion of specific security measures, which have to be achieved taking into consideration the established practices in the implementation of secure communication systems.

5.3.2 Perceived advantages in GSM900/DCS1800 system security.

With regard to the established practices in the implementation of secure communications systems, the GSM system satisfies two of the fundamental criteria, since at no time are any of the cryptographic keys (K_c or K_i) transmitted over the air interface, and all communication of speech, user data and signalling information is encrypted. Since any potential attacker of the system does not know a user's IMSI / K_i pair, he can neither impersonate that user nor eavesdrop upon any conversations.

Thus the four essential user requirements of security of user data, security of signalling information, equipment verification / user authentication and fraud prevention can be satisfied. Note that equipment verification is a matter of cross-referencing the International Mobile Equipment Identity (IMEI) of the mobile station against the Equipment Identity Register (EIR), which is a data base containing lists of approved, non-approved, faulty or stolen equipments. Further considerations regarding the security of the GSM900 PLMN have been made by Rast *et al* [118].

5.3.3 Potential weaknesses in GSM900/DCS1800 system security provisions.

The possible disadvantages and scenarios of attack highlighted below can be considered to be simplified and dependent upon various stated assumptions. They nevertheless constitute the basis of forms of possible attack on the GSM system, which must therefore be seriously considered, since it is imperative that user confidence is maintained in any secure communications system.

Five specific areas of GSM security are considered below, refer to Cooke and Brewster [112] for further consideration.

5.3.3.1 Database security. Since the network implementation of security management is intrinsically linked to data bases, it is essential that these data bases are secure (from unauthorized access for reading or writing data), as are the communications between the data bases and their authorized users. Any successful attack of VLR, HLR or AUC would breach the security of the system, by disclosure of either IMSI / K_i pairs or {RAND, SRES, K_c} authentication sets.

Since the cryptographic algorithms protecting the contents of the above data bases and the communications between them are confidential, it can only be presumed that these algorithms are secure and that security exists in the knowledge of the appropriate cryptographic key alone, and not upon knowledge of the algorithms.

5.3.3.2 Security of algorithm A5 and of ciphering key K_c. Algorithm A5 is the encryption / decryption algorithm for transmitted and received communications traffic. If the plaintext data to be transmitted by a mobile station can be reliably monitored, a known-plaintext attack can be mounted on A5. If there is an ability to reliably monitor the decrypted received communications of the mobile station a chosen-plaintext attack could also be mounted on A5 (similarly possible if the plaintext data to be transmitted by the mobile station can be controlled). A successful cryptanalysis of A5 would yield K_c for the message concerned.

Note that a chosen-plaintext attack upon a cryptosystem allows the cryptanalyst a better probability of breaking a cryptosystem than a known-plaintext attack allows.

Although knowledge of K_c alone (be it gained from a successful attack on A5, VLR or HLR) is only of immediate use for one particular authentication set (that is, for one specific call), its knowledge for that specific authentication set could prove useful to the cryptanalyst (see below).

The ciphering key K_c is 64 bits long and is used in conjunction with a 22-bit TDMA frame number to create the cryptovariable input algorithm A5. The use of the TDMA frame number results in efficient cryptosynchronization between the MS and the BS.

The output of algorithm A5 consists of 114-bit blocks, which are added (modulo-2) to the 114-bit blocks output by the speech coder to perform encryption. The output of algorithm A5 is also used similarly to recover the original input plaintext from the received TDMA frames, the decrypted result of which is subsequently input to the speech decoder at the receiving party.

Assuming that the algorithm A5 is robust the attacker has an initial probability P of successfully recovering the key K_c given by:-

$$P\,(K_c) = 2^{-56} \simeq 5.4 \times 10^{-20} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 5.1$$

Considering this cryptovariable is normally changed at least once during any call, it can reasonably be considered to be secure for the purposes of a PLMN.

5.3.3.3 Security of algorithms A3 and A8, and of identification key K_i. Algorithm A8 is the K_c generation algorithm, which produces a new value of K_c for each value of RAND supplied upon authentication. Algorithm A8 is required to be a one-way function, which would mean that given a value of RAND and its corresponding value of K_c (from above), it would prove to be an intractable problem to calculate the corresponding value of K_i. Since A8 is confidential, it can only be assumed that it has the desired properties of being a truly one-way function; that it is a collision-free hash function.

The same requirement of a robust one-way function exists for algorithm A3, however the threat of attack is much greater here, since the output of the function, SRES, is directly available for cryptanalysis by the attacker.

The RAND value and the K_i each consist of 128 bits and the SRES consists of 32 bits. Thus the total input space S_i for the function is given by :-

$$S_i = 2^{128} \times 2^{128} \simeq 10^{77} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 5.2$$

The output space S_O for the function is given by :-

$$S_O = 2^{32} \simeq 4 \times 10^9 \qquad \text{.......} \qquad 5.3$$

Thus the input space maps onto the output space 2.7×10^{67} times. The reduced output space is used to limit the signalling overhead placed on the air interface, in addition to limiting the probability of successful cryptanalysis of the ciphertext output from the function. The latter is further limited by the response time available for each challenge, since the GSM900/DCS1800 system is bounded by a maximum response time of 500ms; refer to the ETSI GSM recommendations [119].

Due consideration must be given to the possibility of a masquerade attack upon the mobile station. A possible masquerade attack scenario is considered below:-

If a particular mobile subscriber's IMSI or TMSI is known (see below), a masquerade attack can be mounted on the mobile station concerned by means of an illegitimate transceiver local to the mobile station. The mobile station is paged and is supplied with RAND in order for it to reply with its authentication signed response SRES calculated using A3. The algorithm A3 is assumed to be known (it must be assumed that any attacker of a public cryptosystem is able to determine the various cryptographic algorithms concerned; by such means as reverse engineering of the microcontroller within the smart card or by means of industrial espionage).

Any attacker would then know A3, RAND and SRES. Moreover, an attacker would be able to repeat this attack on the mobile station many times with different, *chosen* values of RAND. This would constitute an *unlimited* chosen-plaintext attack on A3. This means that A3 must be a robust one-way function if it is to prove an intractable problem to calculate the mobile subscribers secret cryptographic key, K_i.

It must be emphasized just how important it is that A3 is truly a one-way function; since the algorithm is confidential it can only be assumed that its properties are truly one-way. Any disclosure of K_i would result in a complete breach of security for that particular mobile subscriber, allowing illicit calls to be made, unauthorized connection as the receiving party, and eavesdropping on all calls made and received by the particular mobile subscriber. The latter would be necessarily restricted to the vicinity of the particular cell being used, however tracking the mobile subscriber would be easy in such a circumstance.

5.3.3.4 Security of IMSI and TMSI. In order that the identity of the mobile subscriber is not disclosed by transmissions over the air interface between base station and network, the International Mobile Subscriber Identity (IMSI) is replaced by a Temporary Mobile Subscriber Identity (TMSI), refer to the ETSI GSM recommendations [114]. A new TMSI is transmitted in encrypted form to the mobile station after every occasion on which the mobile station identifies itself, either by originating a call or by being paged by the network.

There are, however, two aspects which should be considered regarding the security of the IMSI / TMSI system of identifying mobile subscribers.

Firstly, no details are available regarding the algorithm which generates TMSI; it should prove to be an intractable problem to calculate IMSI from TMSI.

Secondly, where location registration is performed, since authentication follows identification the mobile station cannot be guaranteed to have a cryptographic key which is commonly known by the VLR. This results in a situation where a mobile station has to supply (or be addressed by) its IMSI, refer to Cooke and Brewster [111], [112]. This situation would also occur if the relation between IMSI and TMSI were lost due to, say, a register failure.

If the above disclosure of IMSI is detected by an eavesdropper, or deliberately invoked by means of a masquerade attack upon the mobile station, it will result in the ability of an attacker to uniquely identify a mobile subscriber and also in the ability to page the mobile subscriber's station by means of the IMSI held by the smart card within it.

5.3.3.5 Security of cryptographic protocols. The cryptographic protocols of the GSM900/DCS1800 PLMN have inherent weaknesses, specifically that at no time is there any authentication of the network entity by the mobile station. This allows impersonation of the network entity by an attacker and similarly allows security-related data being sent from the network to the mobile station to be modified by the attacker.

This problem is demonstrated by formal protocol analysis using the methodology of Rueppel [102], as explained in Section 4.2. The logical productions are shown below; refer to Section 5.2.4.4 for the authentication procedure of the GSM900/DCS1800 PLMN.

$Secu\ (N) \rightarrow sym\ (secu, N), Secu\ (N)$ 5.4

$Secu\ (N) \rightarrow ta\ (secu, N), Secu\ (ta, N)$ 5.5

$Secu\ (ta, N) \rightarrow asym\ (secu, N), ta\ (secu), Auth\ (ta, N)$ 5.6

$Auth\ (ta, N) \rightarrow ta\ (auth, N), Auth\ (N)$ 5.7

$Auth\ (N) \rightarrow phys\ (auth, N)$ 5.8

Production 5.4 shows the classical production of securing *N* data channels by means of a symmetric cryptosystem, requiring the goal of securing *N* (key) channels. There is an individual secret key (K_i) for each channel, the symmetric cryptosystem of which is provided by algorithm A5.

This security goal is achieved in production 5.5 by means of a trusted authority providing security for the *N* data channels, resulting in the new security goal of security for *N* data channels of the trusted authority. The trusted authority will be the AUC/HLR/VLR/MSC of the PLMN.

This security goal is achieved in production 5.6 by means of an asymmetric cryptosystem for securing the *N* secret keys of the *N* data channels and by means of a secure trusted authority; the network is deemed to be intrinsically secure and the cryptosystem is essentially provided for securing communications over the air interface. The asymmetric cryptosystem denotes the one-way hash function (algorithm A3). Note that no session key (K_C) distribution is required in this cryptosystem. The new security goal is deemed to be individual authenticity towards the trusted authority for the *N* data channels.

This security goal is achieved in production 5.7 by means of providing a trusted authority which performs the authentication of each of the *N* data channels, resulting in the new security goal of the secure distribution of *N* secret authentication keys (K_i).

The final security goal is achieved in production 5.8 by means of secure physical distribution of secret authentication keys (K_i) for each of the *N* channels. This is achieved by physical distribution of the individual Subscriber Identity Modules (SIM) for each of the *N* data channels.

Upon parsing the above security architecture it can be seen that there is an error at layer 3. The new security goal of *Auth (ta, N)* (individual authenticity toward the trusted centre) in production 5.6 is deemed to provide *Secu (ta, N)* (distinguished or centralized security); the original security goal. This cannot be guaranteed, since :-

$$Secu\ (ta, N) \rightarrow Auth\ (ta, N),\ Conf\ (ta, N) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad 5.9$$

where:-

- *Secu (ta, N)* denotes distinguished or centralized security, where the trusted centre shares with each of the *N* participants a private channel.
- *Auth (ta, N)* denotes distinguished or centralized authenticity, where the trusted centre has an associated channel from which every party can read, but to which only the trusted centre can write.
- *Conf (ta, N)* denoted distinguished or centralized confidentiality, where the trusted centre has an associated channel to which every party can write, but from which only the trusted centre can read.

This means that an attacker may both modify (authentication) data sent from the trusted centre (the network) to the mobile party, and may impersonate the identity of the trusted centre.

As shown above in Section 5.3.3.3, it is possible to exploit the lack of authentication of the network entity by impersonating a base station for the purpose of paging a mobile station and obtaining the IMSIs of mobile stations within range of the false base station. Another exploitation of this weakness is presented in the following attack scenario :-

- Let an attacker impersonate base station functionality, and be (legitimately or otherwise) in possession of the appropriate identification and authentication data of another mobile subscriber.

- Let the attacker have a directional antenna aligned to the mobile station under attack, with suitable antenna/transceiver gain to mask the legitimate base station in that vicinity.
- Let the false base station perform (standby) location updating procedures with the mobile station under attack.
- Upon occasion of mobile-originated call let (would-be) authentication of the mobile station be performed by the false base station, followed by null-cipher mode being selected by the false base station.
- Let the false base station perform a proper authentication to the local network
- Let the (non-encrypted) call from the mobile station under attack be passed to the network via the (legitimate) mobile station connected to the false base station.

Such an attack would provide full call connectivity between the mobile station under attack and the network, would provide local recovery of all duplex communications between the mobile station under attack and their connected party (upon occasion of mobile-originated calls), and would require no knowledge of security-related algorithms or cryptanalysis of ciphertext.

Such an attack has major implications for the security of the GSM900/DCS1800 PLMN, since it allows potentially unlimited recovery of mobile-originated calls.

5.4 Comparison of the ETSI DECT PLMN system security provisions with those of the ETSI GSM900/DCS1800 PLMN

Table 5.1 summarizes the essential differences between the ETSI DECT PLMN and the GSM900/DCS1800 PLMN. There are few differences between these two systems with regard to implications for system security. There is a subjective difference regarding the accessibility of mobile stations by an attacker insomuch that that DECT has no dedicated control channel. Similarly the control data pertinent to any given mobile station in the GSM PLMN will need to be recovered due to the frequency hopping nature of the mobile channel.

The security architecture of the DECT PLMN is shown in Figure 5.10 below. The variable and algorithm names are explained in Section 5.4.1 below.

5.4.1 DECT Security Services.

There are five security services supported by the ETSI DECT [120], as listed below.

5.4.1.1 Authentication of the Portable Terminal (PT). This security service is initiated by the fixed terminal. It is initiated at the start of a message and at any time thereafter. It is implemented in the network layer.

	DECT	GSM900/DCS1800
Duplex Separation	In time	In frequency 45 Mhz
Multiple Access	12 channel TDMA	8 channel TDMA
Time Frame	12 ms	4.615 ms
Number of Slots	2 x 12 per frame	8 / frame
Speech Codec	32 kbps ADPCM	13 kbps special
Frequency Band	1700 MHz	900 MHz
RF Bit Rate	1024 kbps	271 kbps
Carrier Separation	Approx. 1.3 MHz	200 KHz
Error Protection (speech)	None	r = 0.57
Interleaving (speech)	None	8 frames
Frequency Hopping	None	1 hop / frame
Equalizing	None	16 µs delay
Maximum Peak Power	0.3 W	0.8 W - 5 W hand portable 8 W - 20 W mobile (250 mW - 1 W for PCN)
Range	300 m	32 Km
System Delay (speech)	12 ms	57.5 ms
Dedicated Control Channel	None	Dedicated slots in dedicated carrier
Multiplexing of Signalling and Traffic Data	In each burst	Multi-framing on 26 frames

Table 5.1 Comparison of the RF specifications of the ETSI DECT PLMN and the ETSI GSM900/DCS1800 PLMN

5.4.1.2 Authentication of the Fixed Terminal (FT). This security service is initiated by the PT. It is initiated at the start of a message and at any time thereafter. It is implemented in the network layer.

5.4.1.3 Mutual Authentication. This is a mutual authentication exchange between the PT and the FT.

5.4.1.4 Data Confidentiality. This is requested at the network layer, providing security of user data. It is provided at the MAC layer and is essentially a cryptographic protection for the CAI.

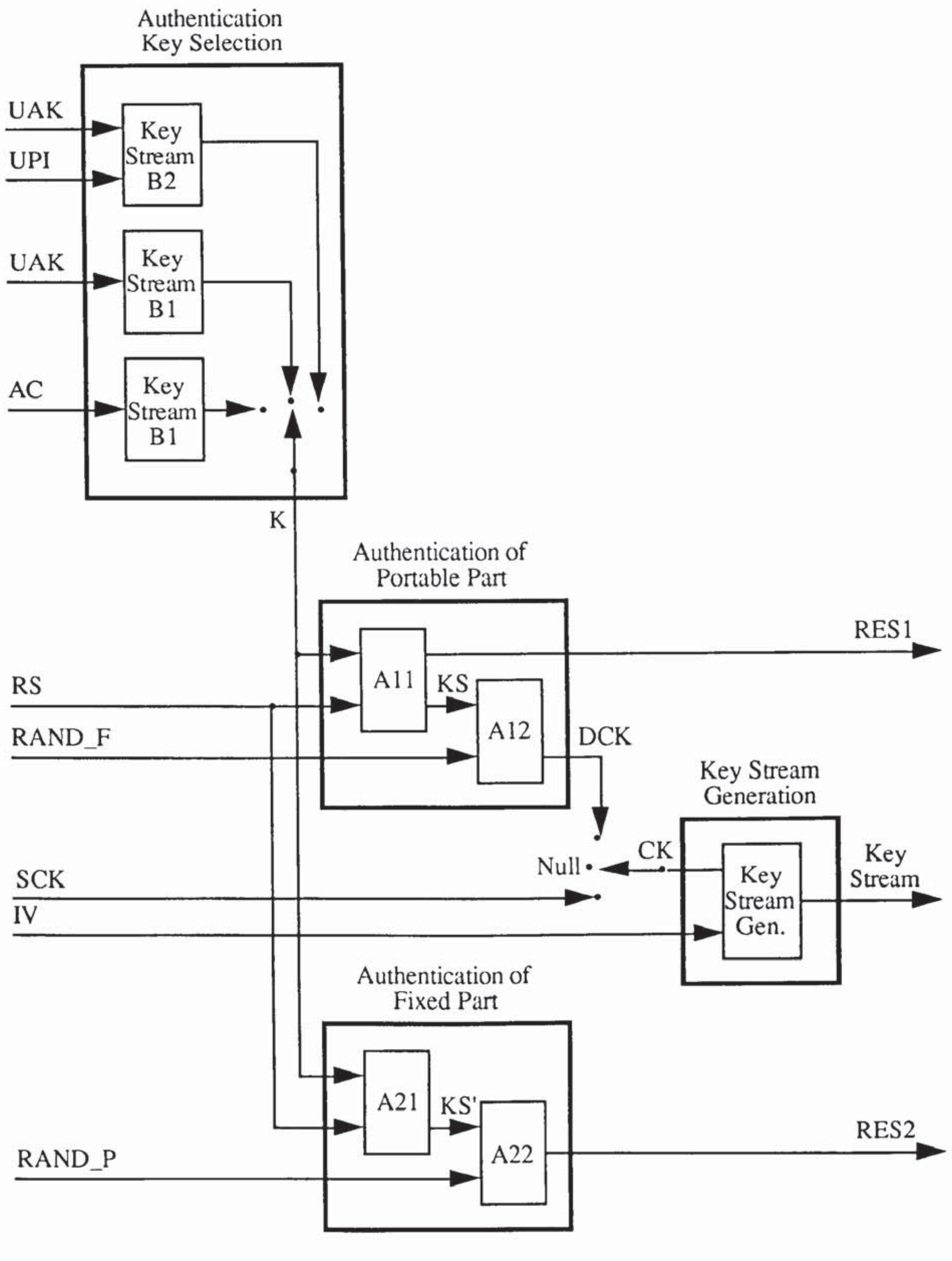


Figure 5.10 Authentication and Ciphering in the ETSI DECT PLMN

5.4.1.5 User Authentication. This is an authentication of the user of a PT by the FT using a User Personal Identity (UPI). It is initiated at the start of a message and at any time thereafter.

5.4.2 DECT Security Mechanisms. There are five security mechanisms supported (or considered for support) by the DECT, as listed below.

5.4.2.1 Authentication of the PT. This is a cryptographic challenge-response exchange, similar to that implemented in the GSM900/DCS1800 PLMN, refer to Walker [121]. The FT obtains values RS, RAND_F and XRES1. RS is the authentication value, RAND_F is a random value and XRES1 is the expected result of authentication. An authentication key, K, is selected and is used in conjunction with RS to create KS, a session authentication key. The calculation is performed using algorithm A11, a one-way function. Similarly XRES1 is calculated from RAND_F and KS using algorithm A12, a one-way function. (Note that calculation of A11 and A12 may be split and may also be performed in advance). The FT then sends RS and RAND_F to the PT.

On receiving the values of RS and RAND_F the PT calculates RES1 and transmits this to the FT.

On receiving the value of RES1 the FT compares it with XRES1 to conform the authenticity of the PT. (Note that KS is a session authentication key which allows the network to support roaming without exposing the authentication key K).

5.4.2.2 Authentication of the FT. This is a cryptographic challenge-response. The PT generates RAND_P, a random number, and sends it to the FT.

The FT obtains RS and RES2 and sends them to the PT. The value of RES2 is obtained from KS' and RAND_P using algorithm A22, a one-way function. KS' is the reverse authentication session key which is obtained from RS and K using algorithm A21, a one-way function. (Note that these calculations may be performed in separate places and that generation of KS' may be performed in advance).

The PT receives RS and RES2 and uses algorithms A21, A22 to calculate XRES2, the expected value of RES2. The latter is then compared with XRES2 to conform the authenticity of the FT. (Note that the visited network is supplied with the authentication set {RS, KS'} from the PT's home system or management centre for each PT. This protects the individual values of K).

5.4.2.3 Mutual Authentication. No explicit mutual authentication mechanism is deemed to be necessary.

5.4.2.4 Data Confidentiality. The common cipher key CK between the FT and the PT allows common key stream generation for bit-wise encryption of communications traffic. The CK may be represented by one of three values: a Derived Cipher Key (DCK), a Static Cipher Key (SCK) or a Null key. The following three points should be noted :-

- DCK is obtained simultaneously with RES1 (or XRES1) from the values of KS and RAND_F using the algorithm A12. Note that authentication of a PT is required at each start of a call and will produce a DCK for the duration of a call. The DCK may be retained by the FT/PT into the next call, until authentication of the PT may be obtained. Authentication of the FT cannot establish a new CK.

- SCK is a static cipher key not generated as above; there is no DECT service for SCK management.

- A null key may be selected by the FT if no ciphering is to be implemented.

5.4.2.5 User Authentication. This is achieved by authentication of a PT where authentication key K is derived from the User Personal Identity (UPI). This value is secret and must be entered every time a user authentication is required. It is entered manually into the PT by the mobile subscriber and generates K. A mechanism is defined to produce K from UPI and a User Authentication Key (UAK), a user authentication key which is part of the mobile subscriber's subscription data. Alternatively an Authentication Code (AC) may be used in place of UAK for temporary PT/FT authentication (see below).

5.4.3 DECT PLMN Cryptosystem Parameters and Keys.

The general parameters of the DECT cryptosystem are given below. Further details are given in the ETSI GSM900 recommendations [120].

AC:	l = 16 to 32 bits May be used to derive K when authentication of PT or of FT service is applied. AC may be held in non-volatile memory in PP or be entered manually by the mobile subscriber for authentication service, depending upon the application. Note that there is no real distinction between UAK and AC, but AC is intended for temporary PT/FT authentication. UAK is more appropriate for long-term authentication.
CK:	l = 64 bits Derived from DCK, SCK or Null.
DCK:	l = 64 bits Derived from RAND_F and KS.

IV:	l = 28 bits Represented by TDMA frame number.
K:	l = 128 bits K is derived from the user-specific DECT keys; UAK, UPI or AC.
KS:	l = 128 bits Derived from RS and K. This authentication session key is provided for authentication of the PT by visited networks, without constant reference to the home network and without disclosing the authentication key K. May be performed at the home network, the local network or at the FT. May be performed in advance of authentication of the PT. Needs to be repeated only if RS or K is changed.
KS':	l = 128 bits Derived from RS and K. Specifications as for KS, except that this authentication session key is provided for authentication of the FT by the PT. Note that if authentication of mobile subscriber and of FT are applied, the value of K for authentication of the mobile subscriber may be derived from UAK and UPI, whilst the value of K used for authentication of the FT may be derived from just UAK.
RAND_F:	l = 64 bits Generated by the FT or within the local network. Generated randomly upon each authentication of the PT.
RAND_P:	l = 64 bits Generated by the PT. Generated randomly (or with low probability of repeating) upon each authentication.
RES1:	l = 32 bits Derived from RS and K.
RES2:	l = 32 bits Derived from RAND_P and KS'. May be compared in the FT or in the local network.

RS:	l = 64 bits Generated by the PT or within the local network. Generally the RS is used for authentication of both the PT and the PT. Note that RS may be the same for several users and/or be used for several authentications.
SCK:	l = 64 bits Held in Portable Part (PP) of PT in non-volatile memory.
UAK:	l is unspecified, but is much greater than the length of AC. May be used to derive K for authentication of PT or FT service. May be used with UPI to derive K with user authentication service. Held in Portable Part (PP) of PT in non-volatile memory.
UPI:	l = 16 to 32 bits May be used to derive K for authentication of PT or FT service. May be used with UPI to derive K with user authentication service. One UAK may have a maximum of one UPI associated with it. UPI is not stored; it is entered into the PT when user authentication is required.
XRES1:	l = 32 bits Derived from RS and K. Comparison may be performed in the FT or in the local network.
XRES2:	l = 32 bits Derived from RAND_P and KS'.

5.5 Evaluation of the ETSI DECT PLMN System Security Provisions

5.5.1 The threat of attack.

As for the GSM900/DCS1800 PLMN, the nature of the threat of attack to the mobile system is necessarily greater than that to the PSTN, since communications between the network and the mobile station are via the air interface, resulting in their ready availability for passive or active attack within the radius of a cell.

Although the cell size is limited to 300m in the DECT, the threat of attack is at least as high due to there being no frequency hopping, thus a dedicated attack upon the system may still prove to be possible. Such an attack might require relatively sophisticated signal processing and cryptanalysis, but it cannot be entirely ruled out.

There are three modes of operation of the DECT PLMN; public access telepoint (PT), business PABX and domestic cordless (DC). The use of the DECT security features in its different modes of operation are defined in Table 5.2 below.

	Pubic Access Telepoint	**Business PABX**	**Domestic Telepoint**
UAK	Applicable	Applicable	Applicable
AC	Applicable only for initial subscription	Applicable	Applicable
UPI/UAK	Applicable only for initial subscription	Applicable only for initial subscription	Not Applicable
SCK	Not Applicable	Applicable	Applicable

Table 5.2 Use of DECT PLMN Security Features

The use of these security features are as anticipated in the ETSI DECT security specifications, [120]. Note that the use of UPI/UAK is deemed to be not applicable in the domestic telepoint implementation of the DECT. The various threats of attack as anticipated in the DECT specification are summarized in Table 5.3 below. Note that the threat due to impersonation of a base station is considered to be high, since any such impersonation may allow unauthorized modification of data in the PT (such as the mobile subscriber's registration data) or a by-pass of the ciphering function. This is considered to be a particularly high threat in the business PABX environment, making authentication of the FT (and not just authentication of the PT) mandatory.

Nature of Attack	**Pubic Access Telepoint**	**Business PABX**	**Domestic Telepoint**
Impersonating a subscriber's identity to avoid call charging	High	High	High
Impersonating a subscriber's identity to provide anonymity	Medium	Medium	Medium
Illicit use of handset (type-approved PP)	Medium	Medium	Medium
Illicit use of handset (non-approved PP)	High	Medium	High
Impersonation of a Base Station (FP)	High	High	High

Table 5.3 Anticipated Threats of Attack for the DECT PLMN

Note that the DECT specification offers no formal support for key management regarding the initial distribution and installation of cryptographic keys, which needs to be performed in a secure manner such that keys cannot be improperly modified or deleted.

The use of session keys provides a distinct advantage since the home network does not have to distribute secret K keys even to authorized visited networks. Such purposes as authentication of PT, authentication of FT and mutual authentication can be fully supported by distribution of authentication triplets {RS, KS, KS'}. Note, however, that the DECT specification does not support a mechanism for authentication triplet distribution.

The actual cipher key CK, whether SCK or DCK, must be available at the point in the FT where encryption is performed. DCK is changed automatically upon each instance of authentication of a PT, but SCK is not. The latter should be changed regularly. Note that the DECT specification does not support management for SCKs.

5.5.2 Perceived advantages in DECT system security.

With regard to the established practices in the implementation of secure communications systems, the DECT system satisfies two of the fundamental criteria, since at no time are any of the cryptographic keys transmitted over the air interface, and all communication of speech, user data and signalling information is encrypted. Note that this status is only afforded if the ciphering mode is selected; this being an anticipated option in the public telepoint and domestic cordless implementations of the DECT.

In this manner the four essential user requirements of security of user data, security of signalling information, equipment verification / user authentication and fraud prevention can be satisfied.

A specific advantage of the DECT PLMN security specification is that it allows an authentication of the FT by the PT, to thwart an attack by impersonating a base station.

5.5.3 Potential weaknesses in DECT PLMN system security provisions.

The possible disadvantages and scenarios of attack highlighted below can be considered to be simplified and dependent upon various stated assumptions. They nevertheless constitute the basis of forms of possible attack on the DECT system, which must therefore be seriously considered, since it is imperative that user confidence is maintained in any secure communications system.

Four specific areas of DECT security are considered below in Sections 5.5.3.1 to 5.5.3.4.

5.5.3.1 Database security. Since the network implementation of security management is intrinsically linked to data bases, it is essential that these data bases are secure (from unauthorized access for reading or writing data), as are the communications between the data bases and their authorized users. As is the case for the GSM900/DCS1800 PLMN, the data base security is taken for granted in this analysis, since any attack is presumed to be made via the air interface.

Since the cryptographic algorithms protecting the contents of the DECT PLMN data bases and the communications between them are confidential, it can only be presumed that these algorithms are secure and that security exists in the knowledge of the appropriate cryptographic key alone, and not upon knowledge of the algorithms. Similar presumptions must be made regarding the nature of the one-way functions (algorithms A11, A12, A21 and A22) and the key stream generator.

5.5.3.2 Security of algorithms {A12, A22, A21, A22} and authentication key K. The RAND_F, RAND_P values each consist of 64 bits, the key K consists of 128 bits and the RES1, RES2 signatures each consist of 32 bits. Thus the total input space S_i for the function is given by :-

$$S_i = 2^{64} \times 2^{128} \simeq 6.3 \times 10^{57} \quad \ldots\ldots\ldots\ldots \quad 5.10$$

The output space S_o for the function is given by :-

$$S_o = 2^{32} \simeq 4 \times 10^{9} \quad \ldots\ldots\ldots\ldots \quad 5.11$$

Thus the input space maps onto the output space 1.6×10^{48} times. The reduced output space is used to limit the signalling overhead placed on the air interface, in addition to limiting the probability of successful cryptanalysis of the ciphertext output from the function. The latter is further limited by the response time available for each challenge; DECT is bounded by a maximum response time of 200ms [120].

Due consideration must be given to the possibility of a masquerade attack upon the PT, by impersonation of a FT (base station). Thus if authentication of the FT is not implemented a possible masquerade attack scenario could be similar to that for the GSM900/DCS1800 PLMN, as shown below. Refer to Cooke and Brewster [111] and [112] :-

Any attacker would know algorithms A11 and A21, RAND_F, RS and RES1. An attacker would be able to repeat this attack on the PT many times with different, *chosen* values of RAND_F and RS. This would constitute an *unlimited* chosen-plaintext attack on A11 and A21. This means that A11 and A21 must be robust one-way functions if it is to prove an intractable problem to calculate the mobile subscribers secret cryptographic key, K. Note, however, that the problems associated with successful cryptanalysis are magnified due to the fact that RES1 is the result of two concatenated one-way functions with different input data.

Assuming that the algorithms {A12, A22, A21, A22} are robust the attacker has an initial probability P of successfully recovering the key K, which is defined by the limits of the key size, thus :-

$$P\ (K) = 2^{-128} \simeq 2.9 \times 10^{-39} \qquad \text{5.12}$$

5.5.3.3 Security of cipher key CK. The probability P of an attacker recovering the cipher key CK (be it derived from either DCK or SCK) for any given message is initially given by :-

$$P\ (CK) = 2^{-64} \simeq 5.4 \times 10^{-20} \qquad \text{5.13}$$

If the cipher key CK is derived from a static cipher key SCK then the key stream value is perturbated by an initial vector IV, represented by the TDMA frame number. Since the TDMA frame numbers change in an incremental manner, they will repeat at known points. Thus to thwart successful cryptanalysis of the transmitted ciphertext the SCK must be changed regularly. Since the length of IV is 28 bits then (at a TDMA duplex repetition rate of 100 frames per second) the IV will repeat every 28 days. This represents the absolute maximum period which should elapse before any SCK is replaced.

5.5.3.4 Security of cryptographic protocols. The protocol analysis of the DECT PLMN is virtually identical to that of the GSM900/DCS1800 PLMN, except that the DECT allows a reverse authentication of the FT. This is essential if impersonation of the FT is to be thwarted.

If authentication of the FT is implemented to thwart impersonation of a base station then the attack scenario proposed for the GSM900/DCS1800 PLMN in which outgoing calls from a mobile station can be recovered and/or modified will no longer be possible.

This problem is demonstrated by formal protocol analysis using the methodology of Rueppel [102], as explained in Section 4.2. The logical productions are shown below; refer to Section 5.4.2 for the authentication procedure of the DECT PLMN.

$$Secu\ (N) \rightarrow sym\ (secu,\ N),\ Secu\ (N) \qquad \text{5.14}$$
$$Secu\ (N) \rightarrow ta\ (secu,\ N),\ Secu\ (ta,\ N) \qquad \text{5.15}$$
$$Secu\ (ta,\ N) \rightarrow asym\ (secu,\ N),\ ta\ (secu),\ Auth\ (ta,\ N),\ Conf\ (ta,\ N) \qquad \text{5.16}$$
$$Auth\ (ta,\ N),\ Conf\ (ta,\ N) \rightarrow ta\ (secu,\ N),\ Auth\ (N) \qquad \text{5.17}$$
$$Auth\ (N) \rightarrow phys\ (auth,\ N) \qquad \text{5.18}$$

Production 5.14 shows the classical production of securing *N* data channels by means of a symmetric cryptosystem, requiring the goal of securing *N* (key) channels. There is an individual secret key (K) for each channel, the symmetric cryptosystem of which is provided by the key stream generator.

This security goal is achieved in production 5.15 by means of a trusted authority providing security for the *N* data channels, resulting in the new security goal of security for *N* data channels of the trusted authority. The trusted authority will be the home and/or visited location registers of the DECT PLMN.

This security goal is achieved in production 5.16 by means of an asymmetric cryptosystem for securing the *N* secret keys of the *N* data channels and by means of a secure trusted authority; the network is deemed to be intrinsically secure and the cryptosystem is essentially provided for securing communications over the air interface. The asymmetric cryptosystem denotes the one-way hash function (algorithms A21, A22 for authentication of the PT and algorithms A21, A22 for authentication of the FT). The new security goal is deemed to be individual authenticity towards the trusted authority for the *N* data channels and also individual confidentiality from the trusted authority for the *N* channels. This is equivalent to distinguished or centralized security; the security goal of production 5.16.

This security goal is achieved in production 5.17 by means of providing a trusted authority which performs the authentication of each of the *N* data channels, and can be authenticated by the *N* data channels. This results in the new security goal of the secure distribution of *N* secret authentication keys (K).

The final security goal is achieved in production 5.18 by means of secure physical distribution of secret authentication keys (K) for each of the *N* channels. This is achieved by physical distribution of the individual DECT Authentication Modules (DAM) for each of the *N* data channels.

Upon parsing the above security architecture it can be seen that the security architecture is logically complete. This is only the case if authentication of both the PT and the FT is achieved.

6.0 Conclusions

This chapter has presented a summary of the security (specifically authentication and encryption) procedures implemented in the ETSI GSM900/DCS1800 PLMN and the ETSI DECT PLMN. The network security architectures have been presented, together with associated protocol diagrams.

A simple analysis of the (initial) probabilities of successful cryptanalysis of the ciphertext transmitted over the air interface by these PLMNs has shown that there is a very low probability of an attacker successfully recovering either a message (ciphering) key or the authentication key of any mobile subscriber. The absence of any detailed specifications pertaining to the algorithms has precluded any academic analysis via modelling and test vectors, however the essential properties of the algorithms are known and have been stated.

This lack of algorithm specifications has not precluded a successful analysis of the cryptosystems concerned, since it has been shown that it is the associated cryptographic protocols which weaken the security afforded by the GSM900/DCS1800 PLMN and may weaken that of the DECT PLMN (depending upon the security options selected).

A formal analysis tool has been used to examine the logic of the cryptographic protocols of the PLMNs and, in the case of of the GSM900/DCS1800 PLMN, has shown that full security can never be provided without authentication of the base station (network). This is an important omission in the security functionality of this PLMN, which has been shown to be provided for in the DECT PLMN.

Thus it has been shown that full authentication needs to be provided before ciphering can be trusted to provide communications privacy.

It is concluded that the difficulty of mounting an (active) chosen-plaintext attack on a mobile station is dependent on both the length of the relevant cryptovariables and the processing time taken to provide a response to a challenge. This will only be the case if the cryptographic algorithms concerned offer the necessary properties.

It is further concluded that the PLMN security architectures are algorithm-independent and thus allow for an easy update of cryptographic algorithms, however the use of proprietary algorithms may limit network inter-operability. This may be further restricted in the DECT PLMN since key management is not part of the common interface specification.

CHAPTER 6. PROPOSALS FOR IMPROVEMENTS TO THE SECURITY PROVISIONS OF SECOND-GENERATION PERSONAL COMMUNICATION SYSTEMS.

This chapter proposes means of enhancing the security functionality of Second-Generation PLMNs, with particular reference to the GSM900/DCS1800 PLMN. The proposed developments ensue from the analysis of the authentication and ciphering security provisions of the Second-Generation PLMNs, as presented in Chapter Four.

Two rationales are adopted in proposing modifications to existing cryptosystem functionality, namely the addition of supplementary security functionality to an existing cryptosystem for the purposes of completing the security architecture such that it is logically complete and the replacement of the original cryptosystem with a new cryptosystem.

The Fiat-Shamir zero-knowledge cryptosystem is examined in detail; the interrelationships between the fundamental parameters of which are analysed and illustrated in a novel manner by means of a graphical technique. The means by which such zero-knowledge cryptosystems may be modified to achieve an optimum solution for a particular PLMN application is demonstrated and is compared with the RSA cryptosystem; emphasis being given to a smart card solution and to the various system overheads concerned.

The processing overheads necessary for the generation of prime numbers for use in asymmetric algorithm cryptosystems are demonstrated by means of a suite of computer programs which model the critical numerical processes.

6.1 Introduction

The previous analysis of the security provisions of Second-Generation PLMN systems (ETSI GSM900/DCS1800 and DECT PLMN) has shown that there are limitations imposed upon the security due to the nature of the cryptographic protocols employed. This chapter explores the possibilities of modifying the cryptosystems to prevent exploitation of protocol weaknesses and also examines the possibility of incorporating different types of cryptographic algorithm, whilst considering the variation in security-related functionality and in overheads.

There are various possible improvements in the security provisions of the GSM900/DCS1800 and DECT PLMNs, particularly to the former. Any security modification would be based upon the specifications of the cryptographic algorithms and/or the cryptographic protocols. The importance of these is broadly considered below.

6.1.1 Algorithm security.

The fundamental properties of any cryptosystem are defined by the type of cryptographic algorithms which are used within the cryptosystem. It is, therefore, essential that all such algorithms are chosen carefully. In any public cryptosystem it is essential that the security of the system lies entirely in the secrecy of the particular cryptographic keys chosen, and not in the secrecy of the algorithms used. Since the algorithms employed in the GSM900/DCS1800 and DECT PLMNs are confidential it can only be presumed that their properties are ideal, and that even if the algorithms were to be obtained (*eg*. by a "reverse engineering" of the integrated circuits contained within mobile stations) their security would not be compromised in any way.

6.1.2 Protocol security.

It is also of prime importance in any cryptosystem that the various cryptographic protocols employed do not allow the security of the algorithms to be compromised; there would be no purpose in having secure algorithms if the cryptographic keys were to be openly transmitted over the air interface.

Given that the Second-Generation PLMN security-related algorithms are deemed to be inherently secure, it is the protocols which relate to them which may prove to be the weak point of the system, this is particularly so with real possibility of masquerade attacks upon the system. Thus it follows that a real possibility for improving the security of such systems exists by employing modified protocols which resist masquerade attacks. Such attacks may be resisted by, for example, limiting the number of authentication requests which may be responded to by a mobile station in a given period of time.

It has been shown in Chapter Five that protocols which ensure a rigourous authentication of both the mobile stations and the host networks are essential.

6.2 Modified PLMN protocols.

It has been shown in Chapter Five that the Second-Generation PLMN systems may be attacked by an impersonation of the base station, where an attacker causes a mobile station to make a false authentication to a false base station and is able to monitor and/or modify calls. Similarly, it has also been shown that an unlimited chosen-plaintext attack may be mounted upon the mobile station in order to attempt to recover the mobile subscriber's secret authentication key.

6.2.1 A modified GSM900/DCS1800 authentication protocol

A modified authentication protocol for the GSM900/DCS1800 PLMN is presented in Figure 6.1 below, which is a development of that proposed by Cooke and Brewster [111], [112]. In this protocol the IMSI is never sent as plaintext across the air interface. When authentication is required the network generates a public key, Kb and a modulus, N. These are then signed using a secret encryption key known only to the network.

The public key Kb and modulus are then sent to the mobile station together with their signatures, E(Kb) and E(N). These are received by all mobile stations within that location area, and the signatures are verified by public decryption keys held in the individual mobile subscriber's SIM (smart card). Only after a successful verification of the authenticity of the parameters Kb and N will they be adopted.

This technique prevents a chosen plaintext attack on the algorithm A. The individual mobile subscriber's IMSI may then be encrypted using algorithm A and returned to the network. In this manner the IMSI is never sent across the air interface as plaintext, providing for complete party anonymity. The encrypted IMSI (IMSI') may then be recovered by the network using its secret decryption key kb.

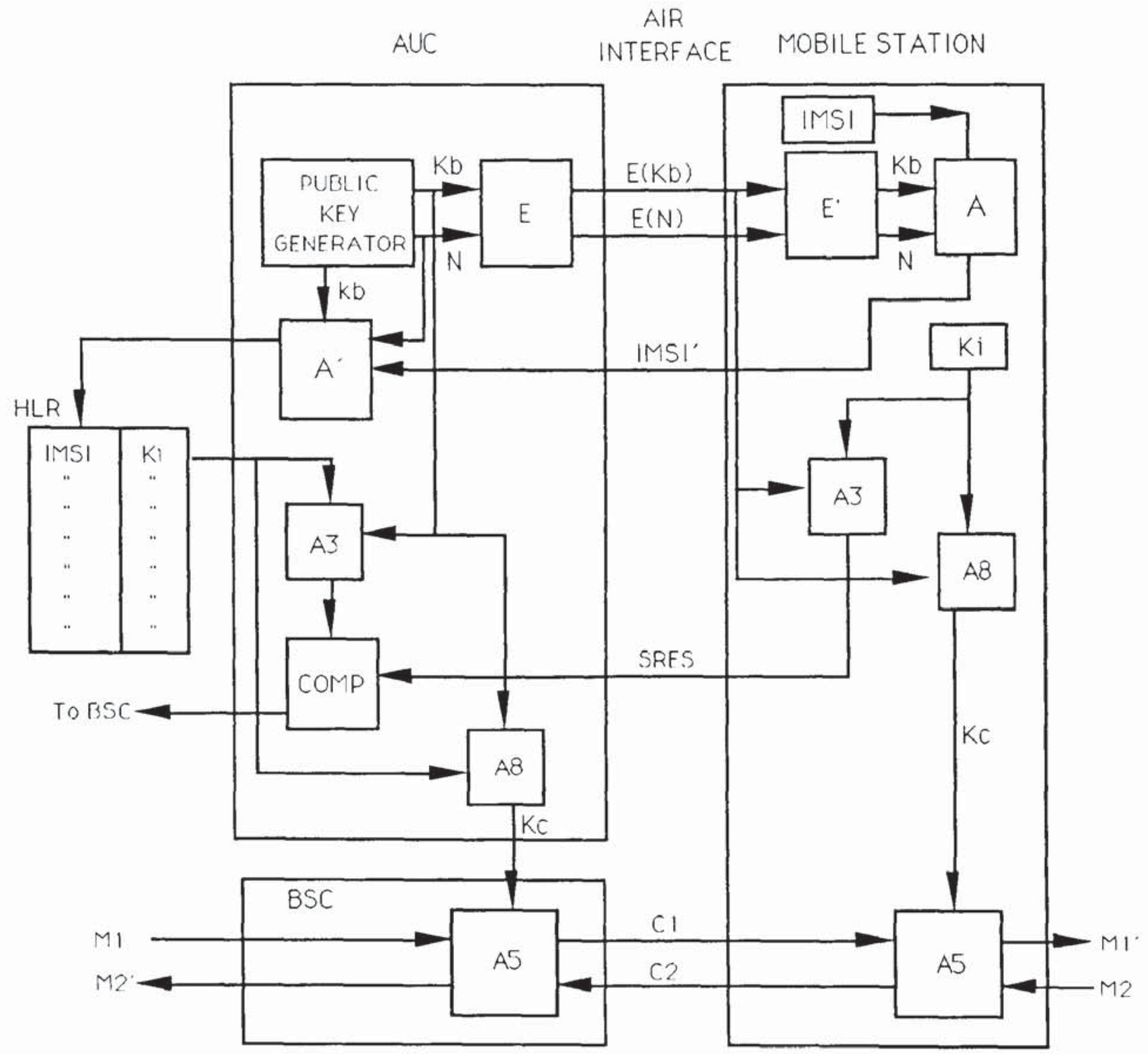


Figure 6.1 Modified GSM900/DCS1800 PLMN Authentication Procedure

The authentication protocol then proceeds as per the original PLMN specification, except that the public key Kb may also be used for perturbation of algorithms A3 and A8, since is is an authenticated random number. This not only provides an economy of signalling information, but also prevents any chosen-plaintext attack on algorithm A3.

The allocation of a TMSI would still be necessary for the purposes of paging the mobile station as it roams throughout the location area, but full authentication will not be necessary. This is because the new TMSI is always allocated in a ciphered manner and an attacker would not be in possession of the message key Kc. This concept is known as implied authentication.

Such a scheme provides continuous encryption, thus protocol reflection attacks are thwarted. Note that there is authentication of the mobile station by the network and authentication of the network by the mobile station. This provides a complete solution to the security architecture, as shown in Chapter Five.

The level of security provided and the overheads to be accommodated are determined by the nature of the cryptographic algorithms employed. These are considered in Section 6.4.

6.2.2 Value added security features in personal communications systems.

Any practical implementation of a secure communications system is bound to be a compromise between the requirements of its users and the various constraints governing its implementation. Thus there may be a situation where a certain group of the users of a system have greater requirements for security and / or functionality of the system, and are willing to pay more for these additional requirements. Thus it would prove beneficial to any public secure communications system if it were to be implemented in such a manner as to allow the ready inclusion of optional extra security features and / or operational facilities.

Such optional extra security features would have to be accommodated by the cryptographic protocols implemented and might include:-

- Cryptographic algorithms providing a higher degree of security (*eg.* public key algorithms). Such algorithms may be user-defined.
- The provision of complete end-to-end security.

The above security options represent a flexible implementation of communications security and would require the use of non-volatile memory in mobile stations and the use of smart cards; the latter being very cheap to produce *en masse*.

6.3 The use of smart cards in personal communication systems security

The use of such an intelligent token to aid the implementation of personal communications security requires it to possess certain functional capabilities, as explained in Cooke and Brewster [122]. These are discussed below.

6.3.1 Overview of the operational and security requirements of intelligent tokens in personal communication systems security.

The limiting factor in implementing cryptosystems using intelligent tokens such as smart cards is neither their communications interface nor their memory capacity, but is purely their maximum processor speed. It is this maximum processor speed which dictates the processing capability available to the cryptographic algorithm to be used, and limits the usable length of cryptographic keys. Thus it is prudent to devise schemes which make efficient use of smart card processor speed, since faster processors are larger, require greater power and cost more to manufacture.

The operational requirements of the intelligent token are twofold; namely that it provides a physical protection of data and a logical protection of data. These operational requirements may be considered to be the ability to provide secure storage of data and to support security-related processing of data. The security requirements of the intelligent token are distinct from those of the communications system to be protected. A fuller description of, and rationale for, these security requirements for the intelligent token are given by Cooke and Brewster [122].

There are three possible mechanisms of authentication associated with an intelligent token, namely authentication of host by card, authentication of token by host and authentication of user by card. These various mechanisms of authentication may be implemented in part or in full to ensure that the host, the token and the user are *bona fide*, respectively.

The protection of information held within, or communicated with, the intelligent token is achieved by means of cryptography. To ensure the robust security of data held within an intelligent token, the data is protected by means of access protocols embedded within the token's microcontroller. These access protocols utilize cryptographic authentication techniques. Similarly, the data communicated with a token may be protected by ciphering. General details of how cryptography may be used in intelligent tokens are given Peyret *et al* [123].

The above constitutes a logical protection of the token's data. Additional physical protection of data held within the token can be provided by the intrinsic nature of the actual silicon chip; all elements of the microcontroller, the memory and the communications interface should be contained within a single slice of silicon. This prevents access to the communications between internal elements of the token.

The parameters held within the smart card may be updated via its communications interface, rather than replacing the card. This may also apply to the algorithms held within the card. This allows the possibility of increased security functionality and near instantaneous updating of user parameters.

The smart card is convenient in size and is cheap enough to be easily replaceable. It is its latter property which permits the easy catering for special subscriber requirements, and allows for the changes necessary to remain in long-term use.

6.3.2 Possible uses of intelligent tokens in personal communication systems security.

There are several uses for intelligent tokens in personal communication systems security, the principal ones are party authentication schemes, the provision of cryptographic algorithms, the provision of security-related data and facility attribution. These are briefly considered below.

6.3.2.1 Party Authentication Schemes. These are distinct from token authentication schemes, as mentioned above. Party authentication schemes may be used at particular nodes in a network at which vulnerability to attack is perceived, and in link end-to-end security where the communicating parties wish to directly authenticate each other. They are typically implemented using digital signature schemes. The threat of attack and types of attack are considered in Cooke and Brewster [112].

6.3.2.2 Provision of cryptographic algorithms. Intelligent tokens are an ideal way of securely providing cryptographic algorithms. Since such tokens provide a high degree of physical and logical protection of data, they can be used to contain algorithms in a secure manner. This provides two security advantages. Firstly, since such tokens are readily replaceable, (particularly if in the form of a smart card), they allow the easy updating of security-related algorithms to provide for future security developments. Secondly, their intrinsic security provides protection of their internal algorithms, (although it is generally bad practice to rely on the secrecy of algorithms; security should exist in knowledge of the cryptographic key alone).

Examples of the provision of cryptographic algorithms by means of intelligent tokens are found in the ETSI GSM900 / DCS1800 [124] and ETSI DECT [120] specifications.

6.3.2.3 Provision of security-related data. Smart cards may be used to hold security-related data. This may include secret identification and authentication data. For example, the actual identity of a mobile subscriber using a public network will not be directly related to the directory number; it will be a completely unrelated number which has been uniquely assigned. This number will be secret and will be held inside the smart card. A secret identification cryptographic key will also typically be held within the smart card, for use in the authentication of a mobile subscriber.

6.3.2.4 Facility Attribution. Another possible use for smart cards in personal communication systems is that of facility attribution. This is where parameters are stored within the smart card, which determine whether certain facilities should be attributed to the owner of the card, such as security services and general services offered by the network operator.

6.3.3 Fundamental aspects of intelligent token-based security schemes.

For the purposes of relevance to the Second-Generation of PLMN systems the intelligent token shall be presumed to be a smart card (that which contains an embedded microcontroller, memory and interface).

A security scheme has to cater for the security requirements of both the mobile subscriber and of the network service provider, and also has to perform within the operational limits due to the nature of the network specifications. The specifications of the mobile station largely determine the operational bounds of such a cryptosystem, due to the necessary limits in their size, cost and power requirement. This gives rise to four essential aspects to be considered when designing such a cryptosystem, namely:-

- The level of security to be provided.
- The mobile station (and smart card) processing overhead.
- The signalling overhead.
- The smart card storage requirements.

These, and other factors, will be considered with regard to implementing smart card security in personal communication systems, with a particular emphasis on zero-knowledge cryptosystems and the RSA public key cryptosystem (as a bench-mark).

6.3.4 Methods of implementing security based on the use of cryptography and intelligent tokens.

The widely accepted bench-mark in comparing the various properties of an asymmetric algorithm cryptosystem is the RSA algorithm [31]. Refer to Section 3.2.1 for an illustration of this scheme.

The security of this cryptosystem is based upon the difficulty of factorizing N, the product of two large primes. There are various means of attempting to break the cryptosystem [37], [38], however if N is large then the number of iterations necessary to break the cryptosystem becomes difficult to manage (for example, if N=664 bits, the number of iterations required = 1.2×10^{23}).

The problem with implementing the RSA cryptosystem is the process intensity required to manage the necessarily large integers. This is particularly the case with smart card implementations; refer to Section 6.3.6. The problem has been alleviated by the advent of zero-knowledge techniques, which have been developed for use in smart card security schemes. These are considered in Section 3.3.

6.3.5 Card-generated and host-generated cryptosystems.

Smart card security schemes can be divided into two major categories with regard to the generation of cryptographic keys; namely card-generated cryptosystems and host-generated cryptosystems.

Card-generated cryptosystems are those in which the smart card performs the necessary calculations for the production of encryption / decryption variables, as opposed to host-generated cryptosystems where the host performs these calculations. The fundamental advantage of the latter category, is that the processing requirement of the smart card is minimized. For the purposes of comparison of different asymmetric algorithms, the typical processing requirement put upon a smart card for the implementation of a host-generated RSA cryptosystem is explained below in Section 6.6.1.2.

6.3.6 Typical smart card specifications.

Smart cards can be considered to be in one of two categories; conventional and specialist. Conventional cards contain a microcontroller linked to memory and an interface, the whole device being fabricated on one slice of silicon. Specialist smart cards have other devices fabricated within them. For the purposes of security such devices can be physical authentication transducers and dedicated cryptographic co-processors.

Typical specifications for conventional smart cards are given below in Table 6.1.

Specialist smart cards are available with dedicated cryptographic co-processors, such as the Siemens™ SLE 44C200 and the Philips™ 83C852.

The Philips™ 83C852 is capable of performing modular exponentiation in 1.5 seconds with 512-bit operands, by virtue of its dedicated cryptographic co-processor.

Parameter	Average Value	Maximum Value
Communications rate (bits/sec)	9600	100, 000
Processor clock frequency (MHz)	4	16
Processor word size (bits)	8	16
RAM size (bytes)	100	512
ROM size (bytes)	4K	20K
EPROM size (bytes)	1K	16K
EEPROM size (bytes)	100	8K

Table 6.1 Typical specifications of conventional smart cards

6.3.7 Current limitations to the use of smart cards in personal communication systems security.

As stated in Section 6.3.1 the only limiting factor in the incorporation of public key cryptography in smart card security is the card processor speed. It is shown in Section 6.5 and Section 6.6 that the processing complexity of a given algorithm is entirely due to the length of the cryptographic keys used. It can thus be seen that any real solution will be a compromise between smart card processor speed and cryptographic key length; the former dictating the cost of the solution and the latter dictating the cryptographic security level.

It can also be seen that, in general, the limiting factors that dictate the functionality of any smart card-based security solution, fall within one of two possible categories: current techniques and current technology. Current techniques include, in particular, the nature of the cryptographic algorithms and protocols used. Current technology refers to the smart card specifications, in particular the processor speed.

Future developments in smart card technology will result in increases in card memory capacity and, more importantly, in the card processor speed. Recently introduced chips, such as the Siemens™ SLE 44C200 and the Philips™ 83C852 incorporate cryptographic hardware capable of performing the computationally intensive calculations required in public key encryption and decryption.

Given that the smart card is ideally suited to the requirements of the intelligent token and that the importance of security in personal communication systems is ever increasing, there can be little doubt that smart cards will play an important role in the security provisions of all future personal communication systems.

6.4 The incorporation of public-key cryptography into communication systems.

6.4.1 The advantages of incorporating public-key cryptography.

There are several advantages to incorporating public-key cryptography into communication systems, the two main ones being advantages in functionality, namely :-

- Secure communications may be achieved from a no-shared-secrets state.
- Link end-to-end security may be achieved between mobile stations.

6.4.1.1 The no-shared-secrets state. The no-shared-secrets state is one in which two or more parties may create a cryptosystem between them for a secure exchange of data, from a state where there is no initially shared secret between the parties. This concept was introduced in Chapter Three.

The significance of this concept is that users can be added to a communications network by an initial exchange of security-related data over a channel which is insecure, without there being any initial common shared secrets and without compromising the security of other users of the system.

6.4.1.2 Link end-to-end security. This facility of protecting communications at all stages between mobile stations, and not just over the air interface, is attracting increasing interest from mobile subscribers. To achieve such security over the whole of the communications path between mobile stations, and guarantee that only the intended parties may recover the protected communications, requires that the communicating parties themselves control their own security.

In realistic terms for a public network, this means that mobile stations would control party authentication and cryptographic key generation. This processing would typically be performed within a mobile subscriber's smart card. Note that only the user-generated data could be protected in this manner, since signalling-related data would be required for routing, paging and billing by the host network(s).

6.4.2 The effects of incorporating public-key cryptography.

It is necessary to assess the effects of the integration of public key cryptosystems into personal communication systems, with regard to the processing requirements put on the smart card, and the signalling and processing requirements put on the network. The inclusion of public key cryptography may well result in an increase in smart card processing requirement, however there are three factors to be considered, apart from the specifications of the public key algorithm, which also determine the overall effect upon the system as a whole. These are hybrid cryptosystems, signalling elimination and distributed processing.

6.4.2.1 Hybrid cryptosystems. The processing and signalling which is required during authentication is only necessarily required upon call set-up or location updating, after a loss (by mobile station or network) of the necessary security-related data. Once a common ciphering key has been established, conventional symmetric algorithm cryptography may be used for ciphering of the communications over the air interface in order to reduce processing requirements. This would constitute a hybrid cryptosystem.

6.4.2.2 Signalling elimination. The introduction of public key cryptography may eliminate the requirement for certain types of signalling, such as the continuous distribution of authentication data to support the ability of the mobile station to roam between different cells and location areas. Refer to Cooke and Brewster [112].

6.4.2.3 Distributed processing. It is possible to distribute some of the processing required to create a public key cryptosystem, between the smart card and the network. Asymmetric algorithm cryptosystems require different amounts of processing by the cryptosystem initiator and the reciprocating party. A typical example of this is the RSA cryptosystem, which is a receiver-generated cryptosystem. The receiver is responsible for generation of one random number and two suitable primes for each cryptosystem generated, in addition to the modular exponentiation of the message to be performed by both parties.

6.5 Incorporating the RSA public key scheme into smart card security schemes.

The processing requirements for RSA public key (asymmetric algorithm) cryptography are large in comparison with conventional (symmetric algorithm) cryptography. By far the most significant processing requirement is the generation of large prime numbers.

6.5.1 Modelling the RSA parameter generation.

Appendix 1 details a suite of programs which have been written to generate primes for the purposes of RSA encryption. The function of this suite of programs is to handle large integers (of size at least 200 bits), to multiply and to divide operands, to perform modular multiplication and to search for prime numbers.

The program PrimeHunter generates the first prime numbers (starting from 3) and stores them in a file SPrimesList. The program ProbEval computes the probability of a number being composite when multiples of the first primes have been suppressed. The results are written to a file ProbList.

The program LargePrimeHunt searches for a prime number of size between MinSize and MaxSize. It first reads the small primes from SPrimesList and stores them into the array SmallPrimes. It then generates a random odd number (Seed) of appropriate size. For every small prime contained in SmallPrimes it initializes a counter to a value equal to the difference between the current value of Seed and the the next multiple of the above primes. Thus the primality of the seed is determined.

If Seed is not prime it is incriminated by two; all counters in SmallPrimes are decremented and tested if any one is zero. Any Seed value which passes the first test for primality must pass a second test using the Solovay and Strassen Method.

The program iterates several times following the algorithm below :-

- Generate a random number A
- Check if GCD(Seed, A) = 1 using function IsGCD1(a, n). If it is false the process is terminated because the Seed is complete.
- Verify whether :-

$$J(A, \text{Seed}) = a(b-1)/2(\text{mod } b) \qquad \text{..} \quad 6.1$$

where J is the function Jacobi, see below.
and a(b-1)/2 is computed using the function PowerMod.

IsGCD1(a, n) is a recursive function returning 1 if GCD(a, n) = 1. The function uses Euler's theorem which states :-

If $\quad a = bn + c$ 6.2

then $\quad \gcd(a, b) = \gcd(b, c)$ 6.3

Function Jacobi avoids the extensive computation involved in evaluating :-

$(-1)(b2 - 1)/8$ and $(-1)(a - 1)(b - 1)/4$

Only the sign of such expressions is required, therefore only the b digits are calculated and are tested for bit 8 = 1.

If Seed passes the SOLOVAY_ITER iterations it is considered to be prime on the basis that the probability of it being composite is less than 1 in 5,000,000 if SOLOVAY_ITER = 20.

The programs were developed for use on a Sun™ *Sparc10* workstation. The results of four sample runs, each of 10 samples, are presented below in Figures 6.2 to 6.5. The samples are for primes of 10, 20, 30 and 40 bytes, respectively.

The spread of the results for generating a prime number is explained by the function Jacobi, which does not have a fixed number of steps. Thus the process time for generating a 40-byte prime number on a Sun™ *Sparc10* workstation can vary from 8 to 23 minutes, (based upon the 10 sample runs presented). The average computation times for the 10, 20, 30 and 40-byte primes are 4.6, 8.6, 13.1 and 16.3 minutes, respectively.

6.5.2 Interpretation of the RSA parameter model results.

A generally accepted minimum size for prime numbers suitable for use in public key cryptosystems is 300 bits. The variation in the processing intensity required to produce prime numbers needs to be considered when implementing such a generator. Such numbers may well be best calculated in advance of their use in a cryptosystem, rather than depending upon real-time generation. Two such primes will be required for each new public key generation.

This need not be a problem, however, since dedicated hardware in the network can continuously produce primes, to be stored for use when required. The problem of generation of random numbers can be readily solved by dedicated hardware. The calculation of the associated secret key for a given pair of public key / prime product, is achieved by solution of the relevant congruence. This (pre-)processing is also able to be performed by central network hardware.

The only processing which must necessarily be performed (in real time) by the smart card at the mobile station is encryption and / or decryption of message data. Both of these operations require the same processing to perform, since they are similar operations.

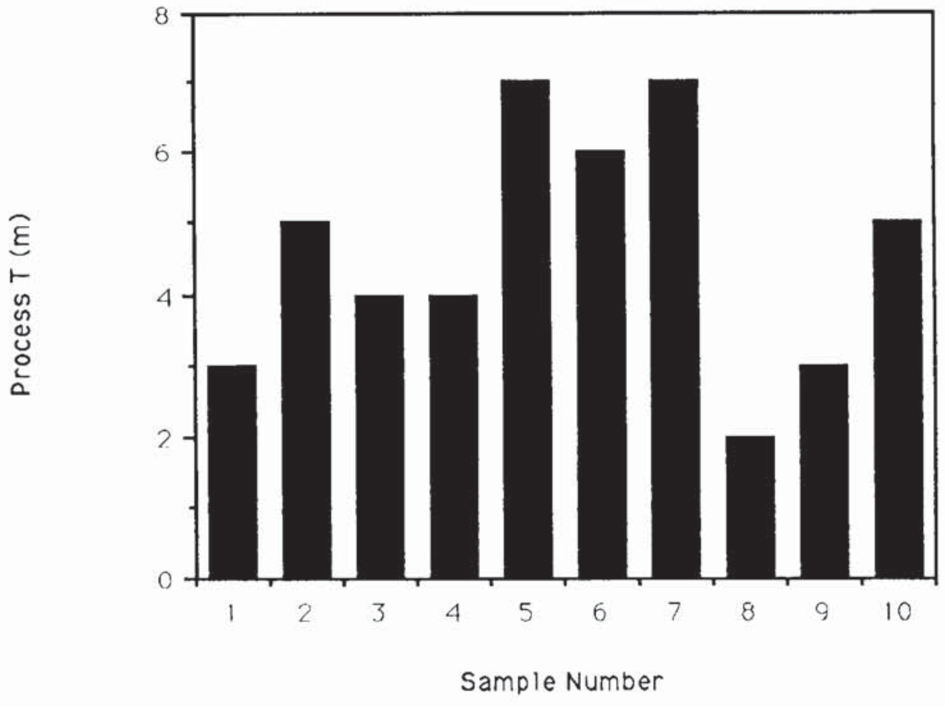


Figure 6.2 10-byte prime number generation times

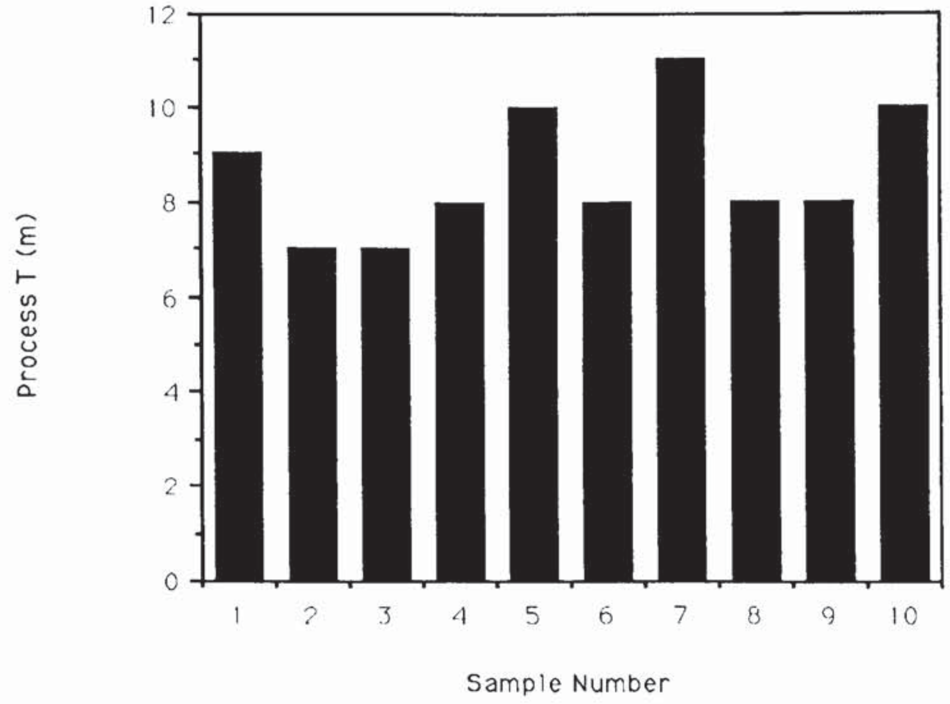


Figure 6.3 20-byte prime number generation times

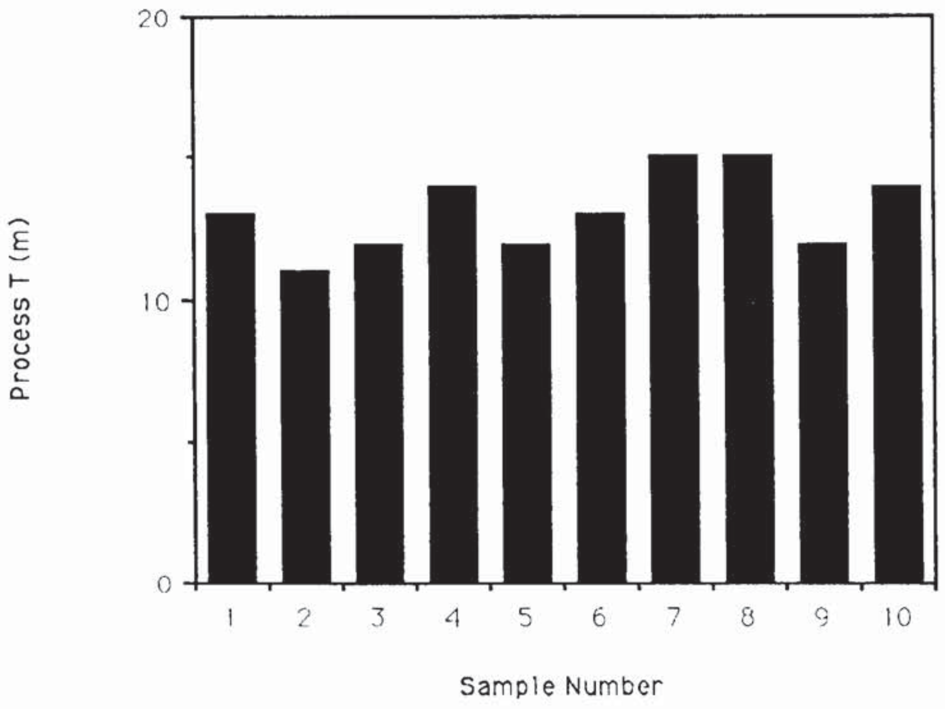


Figure 6.4 30-byte prime number generation times

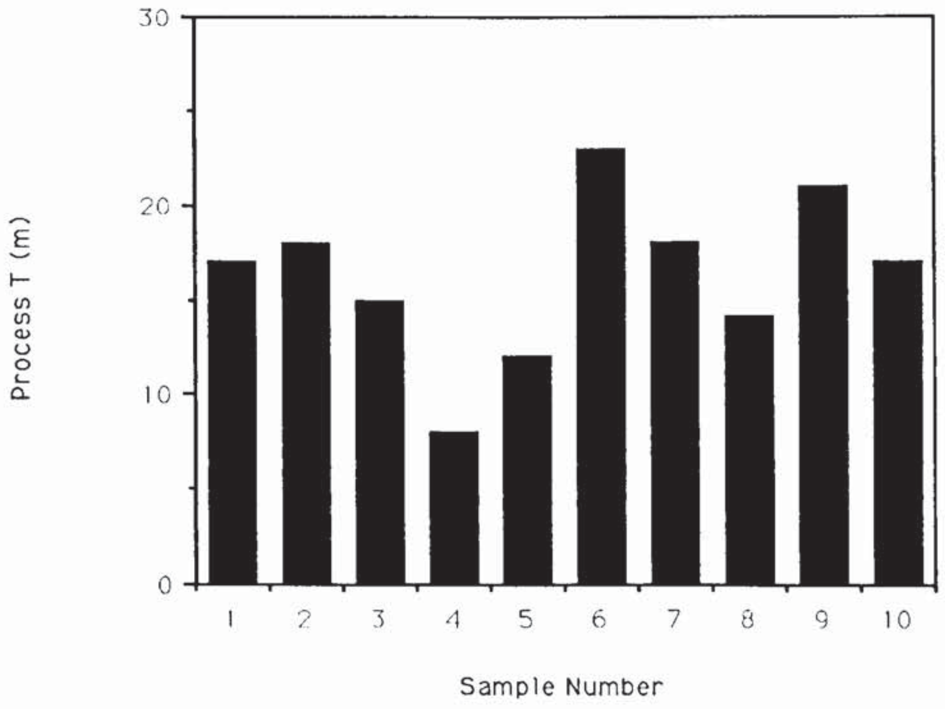


Figure 6.5 40-byte prime number generation times

In the RSA algorithm, the encryption and decryption process can be divided into *n* multiplication and division operations, of which 248 is a typical number. Given a dedicated cryptographic processor of similar specification to the Motorola™ MC 68020 (with a clock frequency = 16 MHz), where the clock period, $\tau = 62.5$ ns :-

MC 68020 (with 32-bit operands):	MULU	=	40 clock cycles
	DIVU	=	108 clock cycles

Thus the multiplication process time T_m for an 80-byte block is :-

$$32^2 \equiv 40 \text{ cycles} \qquad \text{(for a 32-bit operand)}$$
$$640^2 \equiv 16000 \text{ cycles}$$

Thus: $T_m = 16000 \times \tau = 1.0$ ms 6.4

Thus the division process time T_d for an 80-byte block is :-

$$32^2 \equiv 108 \text{ cycles} \qquad \text{(for a 32-bit operand)}$$
$$640^2 \equiv 43200 \text{ cycles}$$

Thus: $T_d = 43200 \times \tau = 2.7$ ms 6.5

Thus to perform encryption / decryption on an 80-byte block, the total processing time T is given by:-

$T = \mathrm{n} \times (\mathrm{T_m} + \mathrm{T_d})$ 6.6

Thus: $T = 248 \times (1 + 2.7) = 917.6 \text{ ms} \approx 1\text{s}$

It can be seen that smart cards with such hardware can be used for real time RSA encryption. It should be noted that the limiting factor in the implementation of such public key cryptosystems on smart cards is the card processor speed. Thus for a given processor speed there will be a compromise between cryptographic key length and security level / processing delay. There is also a trade-off between processor speed and smart card production cost / smart card power requirement. The advent of zero-knowledge techniques has significantly reduced the above necessity to compromise in the design specification of the cryptosystem.

6.6 Incorporating the Fiat-Shamir zero-knowledge authentication scheme into smart card security schemes.

The Fiat-Shamir zero-knowledge cryptosystem is illustrated in Section 3.3. The scheme has been shown to be a good example of zero-knowledge cryptography, since it exhibits a good compromise between the security level provided and the overheads of signalling, processing and storage. Additionally it is very flexible, allowing a large degree of modification of any one of its fundamental parameters to suit a particular application. A further advantage of this scheme is that its variable $c = 2$ (refer to Section 3.3.1). This means that challenges will always be either 0 or 1, and that the modular exponentiation is always to the second power. This substantially reduces the processing overhead placed upon the cryptographic host; refer to Section 6.6.1.2. Further details are given in Cooke and Brewster [125]. A general presentation of smart card cryptographic algorithms is given in [20].

6.6.1 Specifications of the Fiat-Shamir scheme.

The Fiat-Shamir zero-knowledge cryptosystem has the following fundamental specifications :-

Security level $= 2^{-k \times t}$ 6.7

Number of modular multiplications $= t \times (k+2) / 2$ 6.8

Private key storage bits $= k \times n$ 6.9

Signature storage bits $= t \times (n + k)$ 6.10

Communication bits $= t \times (k \times n) / 4$ 6.11

where the globally adjustable parameters are:-

t = number of challenge-response repetitions
k = number of secret integers to be stored
n = number of bits in the modulus

A numerical analysis has been performed regarding the trade-off between the various properties of the Fiat-Shamir cryptosystem. The number of bits in the modulus has been set at 512. This is deemed to provide sufficient security from an attack against the cryptosystem by factorizing n; a composite of two primes. The results of the analysis are summarized below in the form of graphs. Note that only pairs of {k, t} which are factors of k (providing the desired security level) are possible. The results are graphed on two separate pairs of axes for each desired security level, in order to accommodate variations in numerical magnitude.

Note from Figure 6.6 below that, for a constant security level, both the magnitude of the number of modular multiplications and the number of challenge-response repetitions are reduced to an asymptotic minimum as k increases. Similarly it can be seen from Figure 6.7 below that, for the same security level (and the same size modulus) can be achieved whilst simultaneously minimizing the magnitude of both the signature storage and the communicated data. The only compromising parameter is the magnitude of the private key storage.

The presentation of the Fiat-Shamir parameter inter-relationships in this manner allows an immediate inspection of the relationship between any parameters for a given level of security and size of modulus.

As was shown in Section 5.5.3.2, the (initial) security level provided by both the GSM900/DCS1800 and DECT PLMNs is 2^{-128}. To make a comparable system using the Fiat-Shamir algorithm we thus need to make k.t = 128. The parameter relationships for k.t = 128 are shown below in Figures 6.8 and 6.9.

Note from the graphs that, upon comparison of the case where k.t = 72 with the case where k.t = 128, the results are shown in Table 6.2.

It can be seen from the table that, for a minimum of all parameters except the magnitude of the private key storage requirement, all parameters are necessarily increased in magnitude for an increase in the security level.

Parameters (n = 512 bits)	**k.t = 72**	**k.t = 128**
k (number of secret integers to be stored)	72	128
t (number of challenge-response repetitions)	1	1
number of modular multiplications	37	67
number of (mobile) signature storage bytes	73	80
number of (network) private key storage bytes	4608	8192
number of communication bytes	137	160

Table 6.2 Comparison of Fiat-Shamir Parameters, k.t = 72 and k.t = 128

In accommodating such a scheme within a PLMN, the parameters of the cryptosystem need to be considered with regard to the limitations imposed by the PLMN. These parameters are all directly or indirectly affected by the overheads of signalling, processing and storage. There are two premium system parameters which are both overheads, namely the amount of security-related signalling required to be accommodated by the air interface and the amount of processing required to be performed by the mobile station. These are considered below :-

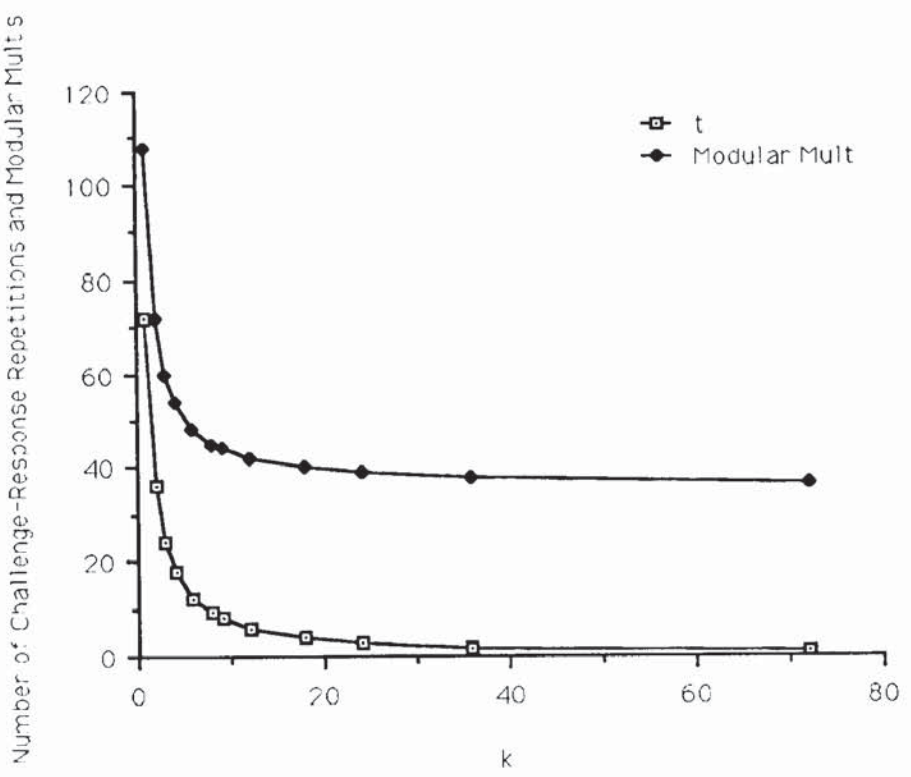


Figure 6.6 Parameter Variation for Fiat-Shamir (kt = 72, n = 512), Number of Challenge-Response Repetitions and Number of Modular Multiplications.

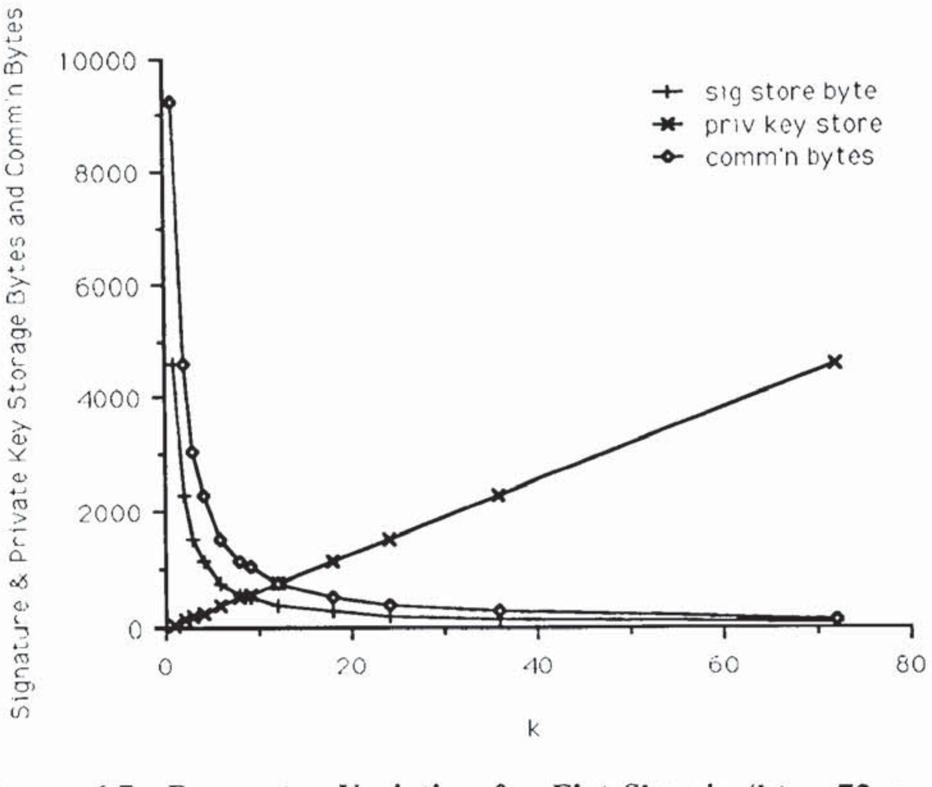


Figure 6.7 Parameter Variation for Fiat-Shamir (kt = 72, n = 512), Number of Bytes Required for Signature Storage, for Private Key Storage and for Communication.

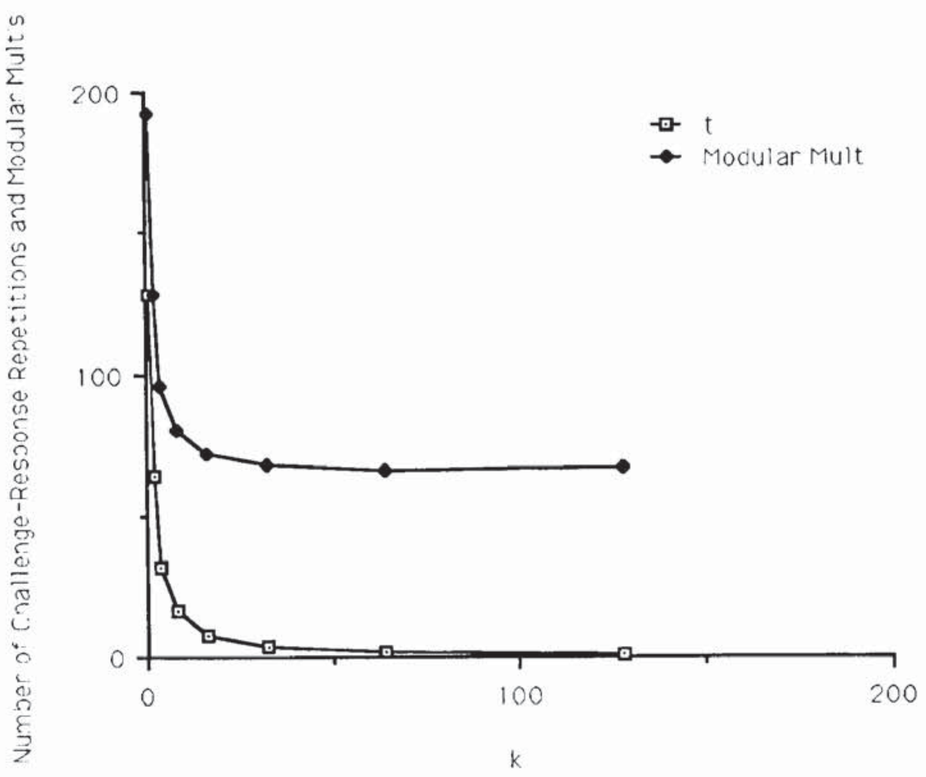


Figure 6.8 Parameter Variation for Fiat-Shamir (kt = 128, n = 512), Number of Challenge-Response Repetitions and Number of Modular Multiplications.

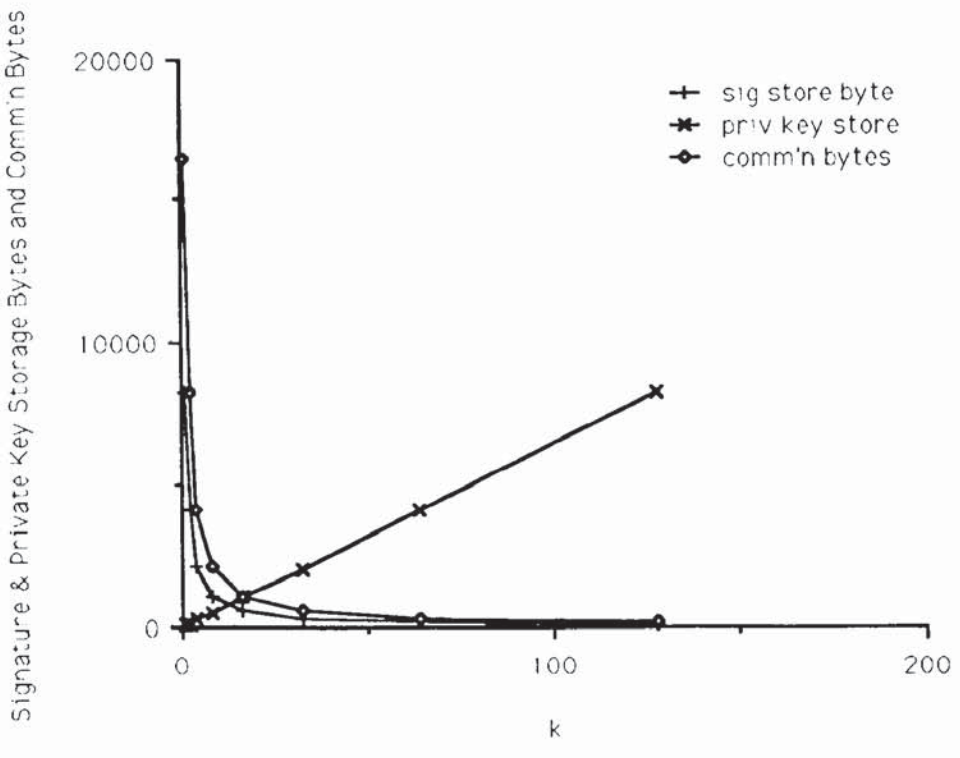


Figure 6.9 Parameter Variation for Fiat-Shamir (kt = 128, n = 512), Number of Bytes Required for Signature Storage, for Private Key Storage and for Communication.

6.6.1.1 Air interface signalling overheads. In comparing the magnitude of the parameters above with the relevant parameters of the GSM900/DCS1800 and DECT PLMNs (refer to Sections 5.3.3 and 5.4.3, respectively) it can be seen that there is a significant increase in the number of transmitted bytes to effect an authentication exchange.

The GSM900/DCS1800 PLMN requires a total number of bytes to be exchanged given by :-

RAND + SRES = 16 + 4 = 20 bytes 6.12

The DECT PLMN requires a total number of bytes to be exchanged given by :-

RS + RAND_F + RES1 = 8 + 8 + 4 = 20 bytes 6.13

This compares with a total of 160 bytes for a Fiat-Shamir zero-knowledge exchange; an eight-fold increase for authentication. An second exchange between network and mobile station would still be necessary to create a common secret ciphering key. Note that a system which uses implied authentication (as explained in Section 6.2) will only require to undertake an authentication when location updating has failed (such as when the mobile station has lost contact with the network due to power-down or poor RF channel) or when the network loses up-to-date information (such as after a register failure).

Thus the overall signalling overhead for authentication is the same after eight occasions where the original PLMN scheme would have undertaken an authentication exchange (upon call set-up and cell hand over).

Furthermore the network signalling overhead is reduced, since there will be a considerable reduction in the amount of security-related data it is required to distribute as the mobile station roams from the domain of one MSC to another. Authentication triplets will only be required when a full authentication is required.

6.6.1.2 Mobile station processing overheads. A comparison of the processing overheads for different cryptosystems can only be performed accurately if the full specifications of the algorithms concerned are available for analysis. This is not the case for the PLMNs concerned, since the algorithms are deemed to be confidential. The upper bound for an authentication response within the PLMN is stated to be 500ms [119].

Note that in the Fiat-Shamir cryptosystem the variable $c = 2$, (refer to Section 3.3.1). This means that the exponentiation is always to the power 2 (mod n). Note, however, that the specifications of the Philips™ 83C852 specialist smart card show that exponentiation with 512-bit operands is possible in 1.5 seconds, refer to Section 6.3.6. This presumes that an RSA-like exponentiation is being performed. It is quite plain that, even using such a processor, the implementation of RSA would be problematic and expensive.

The Fiat-Shamir algorithm, however, only requires that the modular exponentiation is performed to the second power. This is a small fraction of the work required to exponentiate an RSA integer; typically 500 to 10^3 times less work. Thus a Fiat-Shamir exponentiation could be performed in about 1 ms on such a processor. Given the relative ease of performing such a process, there would be no requirement for the cryptographic processing to be specifically minimized at the mobile station; it is possible to select zero-knowledge schemes with different requirements for modular multiplication between the generator of the cryptosystem and the participant.

The authentication exchange would not generate a common secret ciphering key, however the use of zero-knowledge authentication is proposed as a means of allowing an efficient yet secure means of authenticating the mobile party after an initial authentication exchange such as is described in Section 6.2.1. The ciphering key need not be changed upon every occasion for authentication as is required in the GSM900/DCS1800 PLMN; a ciphering key may realistically be used for several messages as is supported in the DECT PLMN.

Note that the challenge-response time is well within the limit stipulated by the PLMN specifications, however there may be cause to artificially increase the response time in order to reduce the probability of successful cryptanalysis via repeated chosen-plaintext attacks if the cryptosystem is such that an attacker can impersonate a base station.

It must be realized that all secure communications systems are a compromise between cost and functionality and that the objective is to provide security over the air interface at a level no less than is afforded by the PSTN. What is important is that security solutions are properly implemented and that the threat of attack is not underestimated.

6.7 Conclusions

The analysis of the security afforded by the GSM900/DCS1800 PLMN and the DECT PLMN made in Chapter Five has demonstrated that there is cause for improving the security schemes of these communications systems.

It has been shown that it is possible to make modifications to these systems such that certain attack scenarios are thwarted. In particular, the impersonation of a network base station has been shown to be avoidable by reverse authentication.

Two rationales have been adopted in proposing changes to PLMN system security functionality:-

- The addition of supplementary security functionality to an existing cryptosystem for the purposes of completing the security architecture such that it is logically complete.
- The replacement of the original cryptosystem with a new cryptosystem.

The implications for the PLMN system of implementing the two approaches have been considered, with particular emphasis on the overheads of signalling, processing and storage.

The Fiat-Shamir zero-knowledge authentication scheme has been examined in detail and it has been shown how its fundamental parameters are inter-related and how they can be optimized to find the best compromise. A novel graphical representation of this has aided the selection of suitable parameters for incorporating the scheme within a PLMN.

The Fiat-Shamir scheme has been compared with the RSA public-key cryptosystem for the purposes of expressing the computational efficiency of the former. The relevance of this has been amplified by illustrating the computational demands of implementing the RSA cryptosystem on a smart card processor. It has been shown how zero-knowledge cryptography can provide some of the useful security functionality of public-key cryptography, but at a fraction of the processing overhead.

The relevance of specialist smart cards has been shown by illustrating the speed with which they can perform the modular multiplications required by asymmetric algorithm cryptography. The rôle of the smart card in PLMN system security has been clearly shown to be essential, and the operational specifications of the smart card have been shown to impose fundamental limitations upon the specifications of the PLMN cryptosystem.

The protocol and signalling limitations imposed by the nature of the PLMN network are shown to be also a limiting factor in imposing fundamental limitations upon the specifications of the PLMN cryptosystem.

The increased security functionality made possible by the adoption of asymmetric algorithm cryptography has been identified and has been shown in the perspective of a PLMN implementation.

The processing overhead put upon the PLMN in adopting asymmetric-algorithm cryptography has been illustrated by modelling the computation necessary in generating suitable prime numbers. A suite of programs has been presented which perform the above function and a selection of results from the model have been given. The results show that the generation of prime numbers can only be realistically performed by the host network due to the intense computational requirements. This limits the implementation of such asymmetric cryptosystems to those which are generated by the network.

As a result of the analysis in Chapter Five of the system security provisions of the GSM900/DCS1800 PLMN, the key factor regarding the implementation of cryptosystems in personal communications systems has been shown to be the nature of the cryptographic protocols employed.

It has been shown that it is essential that any cryptographic algorithms employed in a cryptosystem are implemented using secure protocols which guard against disclosure of cryptographic keys and of party identities, and resist the various forms of attack upon a cryptosystem such as impersonation attacks. It has been shown that it is realistically possible to implement cryptosystems which have secure protocols in a PLMN.

It has been shown that the security solution adopted should provide appropriate security-related functionality and be sufficiently flexible to accommodate the demands imposed by PLMN system inter-compatibility and user operational preference.

CHAPTER 7. A CONSIDERATION OF THE SECURITY REQUIREMENTS OF THE THIRD-GENERATION PERSONAL COMMUNICATION SYSTEMS; UMTS AND FPLMTS.

7.1 Introduction

This chapter contains a preliminary consideration of possible threats to the security of Third-Generation public mobile telecommunications systems. The perceived threats are listed in a categorized manner. There are two reasons for constructing such a list of threats:

- It provides a rationale for the security requirements and security features listed in Sections 7.3 and 7.4, respectively.
- It will assist in a thorough threat analysis of the ETSI UMTS and ITU FPLMTS systems when they become better defined.

The various threats are considered below by reviewing certain areas where it has previously been shown that the security of current Second-Generation systems, such as GSM900/DCS1800 and DECT, could be enhanced. The specific security threats are derived from the generic security requirements of PLMN systems, as specified in Chapter One, and from the specific security weaknesses demonstrably exploited using respective scenarios of attack in Chapter Five. Some of these enhancements are likely to be appropriate to Third-Generation systems. A list of the fundamental threats to the security of the Third-Generation PLMN systems is given below in Section 7.2.

7.2 Threats to Third-Generation Systems

There are a number of possible threats to Third-Generation PLMN systems. These threats may be classified as shown below in Sections 7.2.1 to 7.2.6.

Clearly, many other classifications are possible such as by the threatened party, by the threatening party and by the affected network interface *etc*. The relative usefulness of these various classification schemes would be put into perspective if a threat analysis could be performed on a specific system architecture.

7.2.1 Unauthorized Access to Services

Unauthorized access to services can take place with or without the assistance of an authorized party. This should be remembered when considering each service. Unauthorized access to services can occur in the following ways :-

7.2.1.1 Masquerading as a user. An entity impersonating a legitimate user might utilize services authorized for that user. The entity may have received assistance from other entities such as a service provider or a network operator, or even from the individual user concerned.

7.2.1.2 Masquerading as a subscriber. An entity could impersonate a legitimate subscriber, either to set up a new subscription or to modify a subscription belonging to an existing subscriber. A modified subscription could allow a user to access services not authorized by the respective subscriber.

7.2.1.3 Masquerading as a network operator. An entity could impersonate a network operator (by cloning base stations, satellite receivers or other network equipment) perhaps with the intention of using the access attempts of a legitimate user to gain access to services directly.

7.2.1.4 Masquerading as a service provider. An entity might impersonate a service provider, perhaps with the intention of obtaining the identity of a user or authentication data which could, for example, be used to produce cloned access devices.

7.2.1.5 Misuse of privileges. Entities may abuse their privileges to gain unauthorized access to services. This threat includes misuse of both user and network operator privileges. The subscriber may have to pay the charges incurred by a user who makes unauthorized access to services. A possible example of this is when a network operator offers the user services for which the user is not authorized by the respective subscriber.

7.2.1.6 Theft of terminal equipment and access device. Stolen terminal equipment and access devices could be used to gain unauthorized access to services.

7.2.1.7 Cloning of terminal equipment and access devices. Should an entity obtain an appropriate user identity and authentication data, this could be loaded into fraudulent terminal equipment and access devices resulting in cloning of the originals. These may subsequently be used to gain unauthorized access to services.

7.2.1.8 Non-type-approved equipment. An entity could use a low-cost terminal or other equipment in place of type-approved terminal equipment to utilize services. The threat here is that type approval is circumvented with the possible consequences including adverse effects on the services available to other users.

7.2.2 Denial of Service.

Denial of service occurs when a user is prevented from making use of services to which there exists a legitimate subscription. This can occur in the following ways :-

7.2.2.1 Denial of access. Access to a service could be denied due to inadequate access control. This usually involves intervention by another entity. One example would be a user being locked out of the respectively subscribed account because of repeated, intentionally-failed authentication attempts by another entity. Denial of access could also be accidental (such as a user forgetting the respective PIN) or a side effect of the activity of another entity (such as trying to find out the PIN of another user).

7.2.2.2 Physical intervention. The transfer of messages could be prevented due to inadequate data transfer mechanisms. This usually involves intervention by another entity. One example would be the jamming of signals sent over a radio interface.

7.2.2.3 Protocol intervention. A protocol failure could be made to occur, through the intervention by an entity, causing a service to be terminated. Examples of where problems might occur are hand-over failure and data origin/delivery authentication failure.

7.2.3 Repudiation

Repudiation occurs when an entity falsely denies participation in some action such as the sending or receiving of a message. This can occur in the following ways :-

7.2.3.1 Repudiation of service. A user could deny having attempted to access a service, or deny that the service was actually provided.

7.2.3.2 Repudiation of access to data. A user could deny the action of accessing data, such as the respective service profile or billing data.

7.2.3.3 Repudiation of charge. The repudiation of incurred charges could occur, perhaps because of the above threats.

7.2.4 Unauthorized Access to Data.

This includes the unauthorized disclosure, interception, recovery and cryptanalysis of data. This can occur in the following ways :-

7.2.4.1 Eavesdropping during subscription set-up. There may be eavesdropping on, or interception of, the user/subscriber-associated secret information which passes between the subscriber and service provider when the subscription is set up.

7.2.4.2 Eavesdropping during transmission. Information could be eavesdropped upon when it is transferred between entities over the various interfaces. Examples include an entity using a fake terminal or a terminal with memory, the use of electromagnetic radiation information, the tapping of a communications line or the misuse of a satellite repeater.

7.2.4.3 Disclosure of information whilst accessing terminal equipment with an access device. User or terminal information could be exposed when the user accesses the respective terminal equipment with the respective access device.

7.2.4.4 Disclosure of information by a service provider. User or subscriber information could be exposed by a service provider during auditing, maintenance, back-up and during the provision of services.

7.2.4.5 Disclosure of information via accounting and billing. Insecure accounting and billing practices could result in the disclosure of information. For example, if an access provider receives an itemized bill based on calls initiated from the respective network access then that access provider may also receive the numbers called by registered users. The confidentiality of this data could then be lost. Similarly, the category of billing for calls made could compromise the location confidentiality requirement of the called party.

7.2.4.6 Cryptanalysis of data. User traffic and other encrypted information transmitted over the various interfaces could be intercepted and cryptanalysed.

7.2.4.7 Traffic Analysis. Intruders may observe control data or the lengths, frequencies, sources and destinations of transmitted messages. This may result in the disclosure of confidential information.

7.2.5 Threats to Integrity.

This includes the unauthorized modification, deletion, re-ordering or replay of a valid sequence of communicated messages, as well as the modification or deletion of stored information. The threat may occur in the following ways :-

7.2.5.1 Manipulation of subscription information. Subscription data could be modified or deleted by an entity. This includes the possibility that the entity might modify or delete data by mistake. The result could be that a user is denied access to services, that the entity could gain unauthorized access to services or that the entity could repudiate access and use of services.

7.2.5.2 Manipulation of user information. User identity, authentication and location information could be modified or replayed on the radio path by an entity, possibly with the help of another entity such as a service provider, network operator or even the respective user. The consequences are that either the service is denied to the user or that the service is taken over.

7.2.5.3 Manipulation of user traffic. User traffic could be modified or replayed by an entity.

7.2.5.4 Manipulation of signalling and control information. Signalling and control information could be modified or replayed by an entity.

7.2.6 Other Threats.

7.2.6.1 Unwanted incoming calls to the user. A user may receive unwanted calls. If split charging is used the subscriber has to pay for these calls.

7.3 Security Requirements

There are several security requirements which are for the benefit of users, subscribers and third parties. These security requirements may be considered to be either customer security requirements or service provider security requirements. They include the following :-

- User Access Device.
- Mobile Terminal Equipment.
- Access to Telecommunication Services.
- Charging and Accounting.

The user access device, typically a smart card, is fundamental to the implementation of the network security functionality. It needs to support the required security functionality in addition to providing the necessary level of physical and logical protection of subscriber security-related data.

The mobile terminal equipment needs to support the security functionality required by the network. Consideration is required of the variation in the security functionality which may exist between different service providers and subscribers.

Access to telecommunication services needs to be controlled by both the service provider and the subscriber.

The charging and accounting needs to be controlled only by the service provider.

7.3.1 Security Requirements for UMTS

The requirements for the security features in UMTS is discussed in full in the ETSI Security Principles for UMTS [126]. This list of security requirements has been derived by ETSI/SMG5.

There are additional security requirement for UMTS as specified by SMG5 [126]. They have identified specific supplementary security services to be supported, namely :-

- end-to-end user authentication.
- end-to-end data integrity.
- end-to-end data confidentiality.

The provision of end-to-end security has been generally considered to be an optional extra in the provision of security functionality in all personal communication systems. There are regulatory licensing reasons why such functionality has not been provided in the Second-Generation personal communication systems, which may also restrict such implementation in the Third-Generation systems.

There is a group of users who may desire such increased, user-defined security functionality. Note that only user data may be continuously ciphered between end users; signalling information can only be ciphered between the mobile station and the network, since the signalling data is required by the network for paging, routing and billing.

7.3.2 Security Requirements for FPLMTS.

The latest version of the FPLMTS security requirements are given in the ETSI Security Principles for FPLMTS [127] and are divided into the following categories :-

- service related.
- access related.
- radio interface related.
- terminal related.
- user association related.
- network operational.
- security management.

Each category contains a mixture of security-related system requirements, system requirements on security and security requirements. The security considerations are broadly the same as for UMTS. The security procedures for FPLMTS are given in the ETSI Security

Procedures for FPLMTS [128]. The security procedures are designed to support the necessary functionality to provide for the requirements specified in the Security Principles document [127].

7.4 The Identification of Third-Generation Security Features.

This section discusses which rôles and information should be protected in relation to each of the security feature classes, as summarized below. An in-depth analysis of the security features for Third-Generation PLMNs is presented in the Vodafone™ Technical Report [129].

7.4.1 Confidentiality.

Certain data which is transmitted and stored in Third-Generation environments should be protected against disclosure to unauthorized parties. The data considered here includes signalling data, control data, user traffic data and, in particular, the user identity, the user location, the terminal location, the service profile, charging and billing data and some management information.

The transmitting interfaces considered here include deliveries from a service provider to a user, from a terminal manager to a user, between a user and a network operator, between one network operator and another, between a network operator and a service provider and between one service provider and another.

The storage areas include a user access device or, in UMTS terminology, a User Interface Module (UIM), a terminal, the databases belonging to a service provider and the databases belonging to a network operator.

7.4.2 Integrity.

With regard to confidentiality, any data which is transmitted and stored in Third-Generation environments should be protected in order to meet the needs of integrity. The data considered here includes signalling data, control data, user traffic data and, in particular, the user location, the terminal location, charging and billing data and some management information.

The transmitting interfaces considered here include deliveries from a service provider to a user, from a terminal manager to a user, between a user and a network operator, between one network operator and another, between a network operator and a service provider and between one service provider and another. The storage areas include a UIM, a terminal, the databases belonging to a service provider and the databases belonging to a network operator.

7.4.3 Authentication.

Authentication includes both entity authentication and message authentication. Entity authentication involves several rôles in Third-Generation systems, particularly authentication among users, between service providers and network operators, from terminals and between terminal managers and network operators. Message authentication, as considered here, is message origin authentication, user traffic origin authentication and, in particular, the origin authentication of signalling or control data.

7.4.4 Non-repudiation.

This includes the non-repudiation of data origin and delivery and the non-repudiation of access. The data encompasses signalling data, control data and user traffic. Access encompasses access to stored data and to services.

There may be some argument as to whether these features should be covered under a different heading, such as authentication or access control.

7.4.5 Access Control.

Access control includes access to a facility (e.g. a UIM or terminal equipment), access to a service and access to stored data.

A set of security features has been identified. Each of the features is assigned to one of the categories below.

7.4.5.1 Confidentiality. This includes such security features as the confidentiality of signalling and control data, and the confidentiality of user traffic.

7.4.5.2 Integrity. This includes such security features as the integrity of signalling and control data and the integrity of user traffic.

7.4.5.3 Authentication. This includes such security features as service-related authentication, authentication between network operators and service providers, and message origin authentication.

7.4.5.4 Non-Repudiation. This includes such security features as non-repudiation of origin and delivery of transmitted data, non-repudiation of access to stored data and non-repudiation of access to a service.

7.4.5.5 Access Control. This includes such security features as access control to a facility, to a service and to stored data.

7.4.5.6 Supplementary. This includes such security features as support for the provision of end-to-end security.

7.5 Conclusions

This chapter has presented a brief summary of the (anticipated) security requirements of the future Third-Generation PLMN systems; UMTS and FPLMTS. It has categorized the specific security-related functionality required by Third-Generation service providers and mobile subscribers in terms of the possible threats of attack upon such systems.

The various threats have been considered by reviewing certain areas where it has previously been shown that the security of current Second-Generation systems, such as GSM900/DCS1800 and DECT, could be enhanced. Some of these enhancements are likely to be appropriate to Third-Generation systems. A list of the fundamental threats to the security of the Third-Generation PLMN systems has been given. No attempt has been made to evaluate the individual significance of each of these threats, since the Third-Generation systems have yet to be defined in detail.

It can be seen that such public networks are open to attack in many ways and that comprehensive security functionally is required to address these threats. The implementation of the necessary security mechanisms will require the continued use of cryptography, which may incorporate both symmetric algorithms and asymmetric algorithms. The use of the smart card as a subscriber identity module will be essential in any adopted security solution for such public networks.

It has been shown that the security solution adopted should both provide the appropriate security-related functionality and be sufficiently flexible to accommodate the demands imposed by PLMN system inter-compatibility and user operational preference.

CHAPTER 8. A SUMMARY OF THE CONCLUSIONS REGARDING THE SECURITY ANALYSIS OF PERSONAL COMMUNICATION SYSTEMS AND GENERAL RECOMMENDATIONS FOR THE SECURITY FUNCTIONALITY OF THIRD-GENERATION PERSONAL COMMUNICATION SYSTEMS AND FOR FUTURE RESEARCH.

This chapter presents the major conclusions drawn from the undertaken analysis of cryptographic techniques, as given in Chapters One to Three, the analysis of formal security logic, as given in Chapter Four, the analysis of the security provisions of the Second-Generation personal communication systems, as given in Chapters Five and Six, and the analysis of the security requirements of Third-Generation systems, as given in Chapter Seven.

The conclusions are summarized here and corresponding recommendations are made regarding the provision of security functionality in the future Third-Generation personal communication systems.

Various recommendations are also made regarding the necessary future research perceived in the area of personal communication systems security.

8.1 General conclusions from the security analysis of the Second-Generation personal communication systems.

The analysis of the security provisions of the Second-Generation PLMN systems has consisted of a preliminary investigation of the nature of communications security and of security functionality, followed by a detailed appraisal of the current cryptographic techniques for providing security functionality. The specific security functionality of the Second-Generation personal communication systems (GSM900/DCS1800 and DECT) have been evaluated with regard to the findings of the preliminary investigation. Specific modifications to these systems have subsequently been proposed and evaluated. The major conclusions resulting from the programme of research are as follows :-

8.1.1 Summary conclusions of Chapter One.

This chapter has presented an overview of analogue scrambling schemes and digital encryption schemes, with regard to the generic properties of communication systems and to the threat of attack upon such systems. It has been shown how digital encryption systems can offer the system designer very high security levels for both party authentication and for traffic encryption, in addition to offering the option of extra security-related facilities such as cryptographic key distribution, authentication and data integrity schemes. The essential properties offered by digital cryptosystems are complete flexibility in their implementation and quantitative levels of security. This, in conjunction with their intrinsic compatibility with the digital nature of the Second-Generation PLMN systems, means that the use of digital cryptographic techniques is by far the best means of implementing security in such digital communication systems.

8.1.2 Summary conclusions of Chapter Two.

This chapter has outlined an investigation into the theory pertinent to secrecy systems. It has shown that there are two fundamental properties which the cryptosystem designer must take into consideration when deciding whether to choose a symmetric algorithm in favour of an asymmetric algorithm and when deciding the nature of any symmetric algorithm. Firstly, the designer must assess what effects there will be on the nature of the communication system due to any cryptographic protocols inherent in a cryptosystem which require there to be an initial shared secret, common to communicating parties. Secondly, the designer must assess an appropriate compromise between the level of crypto-secrecy required and the processing and signalling overheads resulting from the use of a particular algorithm in providing the chosen level of crypto-secrecy.

8.1.3 Summary conclusions of Chapter Three.

This chapter has shown how a public key cipher may be used to generate a digital signature scheme and how it may also be used to generate a cryptographic key distribution scheme in relation to the limitations inherent in the use of private key (symmetric algorithm) cryptography.

The importance of the ability to generate both random numbers and prime numbers in such cryptographic schemes has been highlighted and the RSA public key algorithm has been introduced as a typical example of such a cryptosystem.

The Fiat-Shamir scheme for zero-knowledge authentication and digital signature has been presented in comparison to the RSA scheme, demonstrating its computational efficiency. The fundamental issues regarding the practical incorporation of such a cryptographic scheme into a real communication system have been explored in depth. The security levels provided by various cryptosystems have been quantified and have been shown to be dependent upon certain parameters intrinsic to the particular cryptographic algorithm.

Cryptographic key management has been shown to be a major factor for consideration in the implementation of any cryptosystem. Cryptosynchronization has also been shown to be a crucial factor for consideration in the incorporation of any cryptographic scheme into a communication system, together with error propagation. It has been shown how zero-knowledge schemes provide both functional and processing/signalling overhead advantages over general asymmetric algorithm encryption and authentication schemes. This is of particular importance in a mobile radio system where the channel bandwidth and processing power are strictly limited.

Particular significance has been shown in the Fiat-Shamir zero-knowledge scheme, with reference to several proposed modifications. It has also been shown that the cryptographic protocol associated with a given cryptographic algorithm is fundamental to the consideration of the overall signalling overhead.

8.1.4 Summary conclusions of Chapter Four.

This chapter has shown how a formal language of security logic may be used to describe security goals and mechanisms. It has also been shown that such techniques allow security architecture completeness to be proven when the security goals are satisfied. It has been further shown that the management of complexity may be aided by the introduction of a complexity factor into each security goal and security mechanism, allowing such methodologies to be used both as design tools and as evaluation tools.

It has been shown how the BAN approach to protocol security evaluation requires further development to allow for the detection of non-termination of protocols, but that it is unlikely that there is an extension to BAN suitable for the analysis of zero-knowledge protocols.

It has been shown how the rôles of the principals of cryptographic key distribution protocols must be clearly defined before certain types of formal analysis technique, such as BAN, will identify all possible security flaws in all examples of the appropriate types of key distribution protocol.

8.1.5 Summary conclusions of Chapter Five.

In this chapter a formal analysis tool has been used to examine the security logic of the cryptographic protocols of the PLMNs and, in the case of of the ETSI GSM900/DCS1800 PLMN, has shown that full security can never be provided without authentication of the base station (network). This is an important omission in the security functionality of this PLMN, which has been shown to be provided for in the ETSI DECT PLMN.

A lack of algorithm specifications has not precluded a successful analysis of the cryptosystems concerned, since it has been shown that it is the associated cryptographic protocols which weaken the security afforded by the GSM900/DCS1800 PLMN and may weaken that of the DECT PLMN (depending upon the security options selected). Thus it has been shown that full authentication needs to be provided before ciphering can be trusted to provide communications privacy.

It is concluded that the difficulty of mounting an (active) chosen-plaintext attack on a mobile station is dependent upon, and limited to, both the length of the relevant cryptovariables and the processing time taken to provide a response to a challenge. This will only be the case, however, if the cryptographic algorithms and protocols concerned offer the necessary properties.

8.1.6 Summary conclusions of Chapter Six.

This chapter has shown that it is possible to make modifications to the Second-Generation PLMN systems such that certain attack scenarios are thwarted. In particular, the impersonation of a network base station has been shown to be avoidable by reverse authentication.

The implications for the PLMN system of implementing the proposed changes have been considered, with particular emphasis upon the overheads of signalling, processing and storage.

The Fiat-Shamir scheme has been compared with the RSA public-key cryptosystem for the purposes of expressing the computational efficiency of the former. The relevance of this has been amplified by illustrating the computational demands of implementing the RSA cryptosystem on a smart card processor. It has been shown how the Fiat-Shamir scheme can provide some of the useful security functionality of the RSA scheme, but at a fraction of the processing overhead.

The relevance of specialist smart cards has been shown by illustrating the speed with which they can perform the modular multiplications required by asymmetric algorithm cryptography. The rôle of the smart card in PLMN system security has been clearly shown to be essential, and the operational specifications of the smart card have been shown to impose fundamental limitations upon the specifications of the PLMN cryptosystem. The protocol and signalling limitations imposed by the nature of the PLMN network are shown also to be a limiting factor by imposing fundamental limitations upon the specifications of the PLMN cryptosystem.

The processing overhead put upon the PLMN in adopting asymmetric-algorithm cryptography has been illustrated by modelling the computation necessary in generating suitable prime numbers for RSA or Fiat-Shamir. A suite of programs has been presented which perform the above function and a selection of results from the model have been presented. The results show that the generation of prime numbers can only be realistically performed by the host network due to the intense computational requirements. This limits the implementation of such asymmetric cryptosystems to those which are generated by the network.

It has been shown that it is essential that any cryptographic algorithms employed in a cryptosystem are implemented using secure protocols which guard against disclosure of cryptographic keys and party identities, and which also resist the various forms of attack upon a cryptosystem such as impersonation attacks. It has been shown that it is realistically possible to implement cryptosystems in a PLMN which have secure protocols.

8.1.7 Summary conclusions of Chapter Seven.

This chapter has presented a brief summary of the (anticipated) security requirements of the future Third-Generation PLMN systems; UMTS and FPLMTS. It has categorized the specific security-related functionality required by Third-Generation service providers and mobile subscribers in terms of the possible threats of attack upon such systems.

It can be seen that such public networks are open to attack in many ways and that comprehensive security functionally is required to address these threats. The implementation of the necessary security mechanisms will require the continued use of cryptography, which may incorporate both symmetric algorithms and asymmetric algorithms. The use of the smart card as a subscriber identity module will be essential in any adopted security solution for such public networks.

It has been shown that the security solution adopted should both provide the appropriate security-related functionality and be sufficiently flexible to accommodate the demands imposed by PLMN system inter-compatibility and user operational preference.

8.2 General recommendations for the rationale of the development of security functionality in the future Third-Generation PLMN systems.

It is concluded that comprehensive security functionality will be required in future Third-Generation PLMN systems, requiring a high degree of definition and standardization. The gradual evolution of the Second-Generation PLMN systems into the Third-Generation will require that network functionality is both standardized and flexible.

It is recommended that those developing the Third-Generation PLMN systems adopt the following rationale to aide the future development of network security functionality :-

- Portable security functionality is developed, which may be universally incorporated within the Third-Generation PLMN systems.

- The signalling specification of the PLMN networks is considered with a view to all the likely future requirements of security, both of the service provider and of the mobile subscriber.

- The protocol specification of the PLMN networks is considered with a view to all the likely future requirements of security, both of the service provider and of the mobile subscriber.

- All security-related network functionality is developed *ab initio* and is not considered for addition to the network only after demand is demonstrated.

- The final specification of all security-related network functionality is made available for public scrutiny, in the same manner as the rest of the network functional specifications. This is to aid subscriber confidence in the actual security levels provided by such a network. This will also encourage both a common adoption of the standards offered by such a network and further developmental work.

8.3 Recommendations for further research regarding the security functionality of PLMN systems.

There are several areas requiring further research regarding the development of the necessary security functionality for the future Third-Generation PLMN systems, in addition to supplementing and/or replacing the existing security functionality of the current Second-Generation PLMN systems.

These areas for suggested further research are based upon the following considerations :-

- The perceived security and operational requirements of the service provider.

- The perceived security requirements of the mobile subscriber.

- The various progressions which have occurred in cryptographic techniques since the initial development of the Second-Generation PLMN systems.

- The various analyses of the current security functionality of the Second-Generation PLMN systems which have been undertaken since the systems were proposed to the public.

The suggested areas for further research regarding PLMN security are as follows :-

- An in-depth evaluation of the security specification of all proposed zero-knowledge cryptosystems. This would include the security levels achievable, together with the available security functionality.

- An evaluation of the modified system overheads resulting from the adoption of zero-knowledge cryptographic techniques. Such overheads are those of signalling, processing and storage.

- An evaluation of the changes to the existing security-related system overheads after an adoption of new security functionality, such as provided by asymmetric algorithm cryptography.

- An in-depth investigation of the security level achievable by incorporating new cipher schemes such as elliptic curve cryptosystems.

- An evaluation of the new system overheads resulting from the adoption of alternative ciphering techniques. Such overheads are those of signalling and processing.

- A continuing investigation of the functionality offered by the latest smart card technology. The specifications crucial to their incorporation within a cryptosystem are the processing speed (specifically the achievable rate of modular multiplications) and the non-volatile memory capacity.

- An investigation of alternative authentication protocols and cryptographic key distribution schemes. The developments to the (multi-party) certificate system require particular attention.

- A continuing investigation of the optimum security architecture for the host network is required. This requirement is a crucial aspect in the evolution from the Second-Generation PLMN systems to the Third-Generation PLMN systems.

PUBLICATIONS

- Cooke, J.C. and Brewster, R.L. (1992), "Security in Personal Communication Systems", *Fourth Bangor Symposium on Communications*, IEE, pp 172 - 175.
- Cooke, J.C. and Brewster, R.L. (1992), "Cryptographic Security Techniques for Digital Mobile Telephones - Part I", *International Conference on Private Switching Systems and Networks*, IEE, pp 123 - 130.
- Cooke, J.C. and Brewster, R.L. (1992), "Cryptographic Security Techniques for Digital Mobile Telephones - Part II", *IEEE International Conference on Selected Topics in Wireless Communications*, pp 425 - 428.
- Cooke, J.C. and Brewster, R.L. (1993), "The Use of Smart Cards in Personal Communication Systems Security", *IEE Conference on Telecommunications*, pp 246 - 251.
- Cooke, J.C. and Brewster, R.L. (1994), "Cryptographic Algorithms and Protocols for Personal Communication Systems Security", *IEE Colloquium on Security and Cryptography Applications to Radio Systems*, pp 8.1 - 8.6.
- Cooke, J.C. *et al* (1995), "Integration of Zero-Knowledge Cryptography in Personal Communication Systems", *to be submitted.*
- Cooke, J.C. *et al* (1995), "Optimization of Zero-Knowledge Functions by Multi-Dimensional Analysis", *to be submitted.*
- Cooke, J.C. *et al* (1995), "Integration of Cryptographic Security in CT-2 PLMN", *to be submitted.*

REFERENCES

1. Beker, J.B. and Piper, F.C. (1985), "Secure Speech Communications", Academic Press, p 87.

2. Seberry, J. and Pieprzyk, J. (1989), "Cryptography, An Introduction to Computer Security", Prentice Hall, p 280.

3. Pichler, F. (1982), "Analog Scrambling by the General Fast Fourier Transform", *Lecture Notes in Computing Science*, Springer Verlag, v149, pp 173 - 188.

4. Brunner, E., R. (1980), "Efficient Scrambling Techniques for Speech Signals", *Proceedings of the IEEE International Conference on Communications*, pp 16.1.1 - 16.1.6.

5. Beker, J.B. and Piper, F.C. (1985), "Secure Speech Communications", Academic Press, pp 120 - 226.

6. Hong, S.T. and Keubler, W. (1981), "An Analysis of Time Segment Permutation Methods in Analog Voice Privacy Systems", *Proceedings of the Carnahan Conference on Crime Countermeasures*, pp 87 - 94.

7. Jayant, N.S., Cox, R.V., McDermott, B.J. and Quinn, A.M. (1983), "Analog Scramblers for Speech Based on Sequential Permutations in Time and Frequency", *Bell Systems Technical Journal*, v62, n1, pp 24 - 25.

8. Beker, J.B. and Piper, F.C. (1985), "Secure Speech Communications", Academic Press, p 63.

9. FIPS Publication 46 (1977), "Data Encryption Standard", National Bureau of Standards.

10. Diffie, W. and Hellman, M. (1976), "New Directions in Cryptography", *IEEE Transactions in Information Theory*, vIT-22, n6, pp 644 - 654.

11. Rompel, J. (1990), "One-Way Functions are Necessary and Sufficient for Secure Signatures", *Annual ACM Symposium on the Theory of Computing*, pp 387 - 394.

12. Hellman, M.E. (1980), "A Cryptanalytic Time-Memory Trade Off", *IEEE Transactions on Information Theory*, vIT-26, n4, pp 401-406.

13. American National Standard (1985), "American National Standard Data Encryption Algorithm", American National Standards Institution.

14. Shannon, C.E. (1949), "Communication Theory of Secrecy Systems", *Bell Systems Technical Journal*, v28, pp 656 - 715.

15. Davies, D.W. and Price, W.L. (1984), "Security for Computer Networks", John Wiley and Sons, p 37.

16. Gibson, J.D., Stanners, S.P. and McLellan, S.A. (1993), "Spectral Entropy and Coefficient Rate for Speech Coding", *27th Asilomar Conference on Signals, Systems and Computers*, ch329, pp 925 - 929.

17. Bluhme, H. (1987), "Entropy and Redundancy in Text and Language", *International Conference on Communications Technology*, (Published by *World Scientific Publishing Co. Inc.*), ch. 253, pp 466 - 472.

18. Vantilburg, J. and Boekee, D.E. (1985), "Divergence Bounds on Key Equivocation and Error Probability in Cryptanalysis", *Advances in Cryptology: Crypto 85*, Lecture Notes in Computer Science, Springer-Verlag, ch. 44, pp 489 - 513.

19. Damours, C., Chouinard, J.Y. and Yongacoglu, A. (1993), "Unicity Distance of Linear and Nonlinear Pseudonoise Sequence Generators for Direct-Sequence Spread-Spectrum Systems", *Proceedings of the IEEE Global Conference on Communications; Globecom '93*, ch. 400, pp 159 - 163.

20. Königs, H.-P. (1991), "Cryptographic Identification Methods for Smart Cards in the Process of Standardisation", *IEEE Communications Magazine*, June 1991, pp 42 - 48.

21. Seberry, J. and Pieprzyz, J. (1989), "Cryptography, An Introduction to Computer Security", Prentice Hall, pp 78 - 86.

22. Brown, L. (1987), "A Software Implementation of the DES in 'C'", *Internal Report*, Australian Defence Force Academy, University of New South Wales, Australia.

23. Shimizu, A. and Miyaguchi, S. (1987), "Fast Data Encipherment Algorithm, FEAL", *Advances in Cryptography: Eurocrypt 87*, Springer-Verlag, pp V1 - V11.

24. ETSI, (1990), "European Digital Cellular Telecommunications System", GSM, prI-ETS 300 020.

25. ETSI, (1991), "Digital European Cordless Telecommunications Common Interface", DECT, pt. 1: Overview, ETSI, prETS 300 175 - 1.

26. Davies, D.W. and Price, W.L. (1984), "Security for Computer Networks", John Wiley and Sons, pp 264 - 265.

27. Pichler, F. (1987), "Finite State Machine Modelling of Cryptographic Systems in Loops", *Advances in Cryptography: Eurocrypt 87*, Springer-Verlag, ch. 26, pp 65 - 73.

28. Needham, R.M. and Schroeder, M.D. (1978), "Using Encryption for Authentication in Large Networks of Computers", *Communications of the ACM*, v21, n12, pp 993 - 999.

29. Miller, S.P., Neuman, B.C. and Saltzer, J.H. (1987), "Kerberos Authentication and Authorization System", *MIT Project Athena*, Cambridge, Ma.

30. Garey, M.R. and Johnson, D.S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness", W.H. Freeman and Son, p 154.

31. Rivest, R., Shamir, A. and Adleman, L. (1978), "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", *Communications of the ACM*, v21, n2, pp 120 - 128.

32. Odlyzko, A.M. (1984), "Discrete Logarithms in Finite Fields and Their Cryptographic Significance", *Advances in Cryptography: Eurocrypt 84*, Springer-Verlag, pp 224 - 314.

33. Seberry, J. and Pieprzyz, J. (1989), "Cryptography, An Introduction to Computer Security", Prentice Hall, p92 - 93.

34. Blakeley, G.R. and Borosh, I (1987), "Rivest-Shamir-Adleman Public-Key Cryptosystems do not Always Conceal Messages", *Computers and Mathematics with Applications*, v5, pp 169 - 178.

35. Fiat, A. and Shamir, A. (1987), "How to Prove Yourself: Practical Solutions to Identification and Signature Problems", *Advances in Cryptology: Crypto 86*, Lecture Notes in Computer Science, Springer-Verlag, pp 186 - 194.

36. Seberry, J. and Pieprzyz, J. (1989), "Cryptography, An Introduction to Computer Security", Prentice Hall, p278.

37. Denning, D.R. (1982), "Cryptography and Data Security", Addison-Wesley.

38. Simmons, G.J. and Norris, M.J. (1987), "Preliminary Comments on the MIT Public Key Cryptosystem", *Cryptologia*, v1, pp 406 - 414.

39. Rivest, R.L. (1979), "Critical Remarks on 'Critical Remarks on Some Public-Key Cryptosystems', by Herlestam, T.", *BIT*, v19, April 1979, pp 274 - 275.

40. de Brito, G. (1988), "Low Bit Rate Speech for the GSM System", *IEEE Eurocon Conference on Electrotechnics*, pp 19 - 23.

41. Huber, K. (1991), "Some Considerations Regarding the Selection of RSA Moduli", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 294 - 301.

42. Sun, H.-M. and Hwang, T (1991), "Public-Key Identity Based Cryptosystem", *IEEE Carnahan Conference on Security Technology*, pp 142 - 143.

43. Chang, C.-C. and Hwang, R.-J. (1981), "A Strategy for Transforming Public Key Cryptosystems into Identity-Based Cryptosystems", *IEEE Carnahan Conference on Security Technology*, pp 68 - 72.

44. Dåmgard, I. (1991), "Towards Practical Public Key Cryptosystems Secure Against Chosen-Ciphertext Attacks", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 445 - 456.

45. Fiat, A. (1989), "Batch RSA", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 175 - 185.

46. Hoornaert, F., Decroos, M., Vanderwalle, J. and Govaerts, R. (1988), "Fast RSA-Hardware: Dream or Reality", *Advances in Cryptography: Eurocrypt 88*, Springer-Verlag, pp 257 - 264.

47. Gallay, P. and Depret, E. (1988), "A Cryptography Processor", *IEEE International Solid State Circuits Conference*, pp 148 - 150.

48. Kameyama, M., Wei, S. and Miguchi, T., (1990), "Design of an RSA Encryption Processor Based on Signed-Digit Multivalued Arithmetic Circuits", *Systems and Computing in Japan*, v21, n6, pp 21 - 31.

49. Bender, A. and Castagnoli, G. (1989), "On the Implementation of Elliptic Curve Cryptosystems", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 186 - 192.

50. Koblitz, N (1990), "Constructing Elliptic Curve Cryptosystems in Characteristic-2", *Advances in Cryptology: Crypto 90*, Lecture Notes in Computer Science, Springer-Verlag, pp 156 - 167.

51. Koyama, K., Maurer, U.M., Okamoto, T. and Vanstone, S.A. (1991), "New Public Key Schemes Based on Elliptic Curves Over the Ring Zn", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 252 - 266.

52. Koblitz, N. (1991), "CM-Curves with Good Cryptographic Properties", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 279-287.

53. Beth, T. and Schaefer, F. (1991), "Non-Supersingular Elliptic Curves for Public Key Cryptosystems", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 316 - 327.

54. El Gamal, T. (1985), "A Public Key Cryptosystem and Signature Scheme Based on Discrete Logarithms", *IEEE Transactions in Information Theory*, vIT-31, n4, pp 467 - 472.

55. Menzes, A. and Vanstone, S. (1990), "Implementation of Elliptic Curve Cryptosystems", *Lecture Notes in Computing Science*, v453, Springer-Verlag, pp 2 - 13.

56. Agnew, G.B., Mullin, R.C. and Vanstone, S.A. (1989), "A Fast Elliptic Curve Cryptosystem", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 706 - 708.

57. Chang C.C. and Lin, C.H. (1989), "An Identity-Based Signature Scheme Based Upon Rabin's Public Key Cryptosystem", *IEEE International Carnahan Conference on Security Technology*, pp 139 - 141.

58. Rabin, M.O. (1979),"Digitalized Signatures and Public Key Functions as Intractable as Factorization", *Technical Report*, MIT/LCS/TR212, MIT Laboratories, Department of Computer Science, Cambridge, Mass.

59. Desmedt, Y. and Frankel, Y. (1991), "Shared Generation of Authentication and Signatures", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 457 - 469.

60. Chaum, D., van Heijst, E. and Pfitzmann, B. (1991), "Cryptographically Strong Undeniable Signatures, Unconditionally Secure for the Signer", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 470 - 484.

61. Chaum, D. and Roijakkers, S. (1990), "Unconditionally Secure Digital Signatures", *Advances in Cryptology: Crypto 90*, Lecture Notes in Computer Science, Springer-Verlag, pp 206 - 214.

62. Galil, Z., Haber, S. and Yung, M. (1989), "A Secure Public Key Authentication Scheme", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 3 - 15.

63. Ohta, K. and Koyama, K. (1991), "Meet-in-the-Middle Attack on Digital Signature Schemes", *Lecture Notes in Computing Science 1990*, v453, pp 140 - 154.

64. Fujioka, A., Okamoto, T. and Miyaguchi, S (1991), "ESIGN: An Efficient Digital Signature Scheme for Smart Cards", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 446 - 457.

65. Stern, J. (1989), "An Alternative to the Fiat-Shamir Protocol", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 173 - 180.

66. Ohta, K. and Okamoto, T. (1988), "A Modification of the Fiat-Shamir Scheme", *Advances in Cryptology: Crypto 88*, Lecture Notes in Computer Science, Springer-Verlag, pp 232 - 243.

67. Brandt, J., Dåmgard, I, Landrock, P. and Pedersen, T. (1988), "Zero Knowledge Authentication Scheme with Secret Key Exchange", *Advances in Cryptology: Crypto 88*, Lecture Notes in Computer Science, Springer-Verlag, pp 583 - 588.

68. Beth, T (1988), "Efficient Zero-Knowledge Identification Scheme for Smart Cards", *Advances in Cryptography: Eurocrypt 88*, Springer-Verlag, pp 77 - 84.

69. Knobloch, H.-J. (1988), "A Smart Card Implementation of the Fiat-Shamir Identification Scheme", *Advances in Cryptography: Eurocrypt 88*, Springer-Verlag, pp 87 - 95.

70. Guillou, L.C. and Quisquater, J.-J. (1988), "A Practical Zero Knowledge Protocol Fitted to security Microprocessor Minimising both Transmission and memory", *Advances in Cryptography: Eurocrypt 88*, Springer-Verlag, pp 123 - 128.

71. Schnorr, C.P. (1989), "Efficient Identification and Signatures for Smart Cards", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 239 - 251.

72. de - Rooij, P. (1991), "On the Security of the Schnorr Scheme Using Preprocessing", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 71 - 80.

73. Shamir, A. (1989), "An Efficient Identification Scheme Based on Permuted Kernels", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 606 - 609.

74. Haber, S. and Stornetta, W.S. (1990), "How to Time Stamp a Digital Document", *Advances in Cryptology: Crypto 90*, Lecture Notes in Computer Science, Springer-Verlag, pp 437 - 455.

75. Okamoto, E. (1986), "Proposal for Identification-Based Key Distribution Systems", *Electronics Letters*, n22, pp 1283 - 1284.

76. Koyama, K. and Ohta, K. (1988), "Security of Improved Identification-Based Conference Key Distribution Systems", *Advances in Cryptography: Eurocrypt 88*, Springer-Verlag, pp 11-19.

77. Tatebayashi, M., Matsuzaki, N. and Newman Jr., D.B. (1989), "Key Distribution Protocol for Digital Mobile Communication Systems", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 324 - 334.

78. Yacobi, Y. and Shmuely, Z. (1989), "On Key Distribution Systems", *Advances in Cryptology: Crypto 89*, Lecture Notes in Computer Science, Springer-Verlag, pp 344 - 355.

79. Günther, C.G. (1989), "An Identification-Based Key Exchange Protocol", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 29 - 37.

80. Shamir, A (1985), "Identification-Based Cryptosystems and Signature Schemes", *Advances in Cryptology: Crypto 84*, Lecture Notes in Computer Science, Springer-Verlag, pp 47 - 53.

81. Davida, G., Desmedt, Y. and Peralta, R (1989), "A Key Distribution System Based on a One-Way Function", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 75 - 79.

82. Ferrer, J.D. and Rotger, L.H. (1989), "Full Secure Key Exchange and Authentication With No Previously Shared Secrets", *Advances in Cryptography: Eurocrypt 89*, Springer-Verlag, pp 665 - 669.

83. Tsujii, S. and Chao, J. (1991), "A New Identification-Based Key Sharing system", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 288 - 299.

84. Maurer, U.M. and Yacobi, Y. (1991), "Non-Interactive Public Key Cryptography", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 498 - 507.

85. Fumy, W. and Munzert, M. (1990), "A Modular Approach to Key Distribution", *Advances in Cryptology: Crypto 90*, Lecture Notes in Computer Science, Springer-Verlag, pp 274 - 283.

86. Proctor, N. (1985), "A Self-Synchronizing Cascaded Cipher System with Dynamic Control of Error Propagation", *Advances in Cryptology: Crypto 84*, Lecture Notes in Computer Science, Springer-Verlag, ch. 35, pp 174 - 190.

87. Daemen, J, Govaerts, R. and Vandewalle, J. (1992), "On the Design of Self Synchronizing Stream Ciphers", *Proceedings of ISITA '92*, pp 16 - 20.

88. Berger, R., Kannan, S. and Peralta, R. (1985), "A Framework for the Study of Cryptographic Protocols", *Advances in Cryptology: Crypto 85*, Lecture Notes in Computer Science, Springer-Verlag, pp 87 - 103.

89. Rabin, M.O. (1976), "Probabilistic Algorithms", *Algorithms and Complexity*, Academic Press, New York, pp 21 - 40.

90. Hauser, R.C. and Lee, E.S. (1992), "Verification and Modelling of Authentication Protocols", *IEEE Symposium on Research and Security*, pp 141 - 154.

91. Burrows, M., Abadi, M. and Needham, R. (1989), "A Logic of Authentication", *Operating Systems review*, v23, n5, pp 1 - 13.

92. Steiner, J.G., Neuman, C. and Schiller, J.I. (1988), "Kerberos: An Authentication Service for Open Network Systems", *Proceedings of Usenix Winter Conference*, 1988.

93. Denning, D.E. and Sacco, G.M. (1981), "Timestamps in Key Distribution Protocols", *Communications of the ACM*, v24, p533.

94. Bauer, R.K., Berson, T.A. and Feirtag, R.J. (1983), "A Key Distribution Protocol Using Event Markers", *ACM Transactions on Computer Systems*, v1, n3, pp 249 - 255.

95. Syverson, P (1991), "The Use of Logic in the Analysis of Cryptographic Protocols", *IEEE Symposium on Security and Privacy*, pp 156 - 170.

96. Daemen, J, Govaerts, R. and Vandewalle, J. (1992), "A Hardware Design Model for Cryptographic Algorithms", 2nd European Symposium on Research in Computer Security - ESORICS '92, Springer-Verlag, Lecture Notes in Computing Science, ch. 24, pp 419 - 434.

97. Abadi, M., Burrows, M., Kaufman, C. and Lampson, B (1991), "Authentication and Delegation with Smart Cards", *TACS 1991*, Springer-Verlag, Lecture Notes in Computing Science, v526, pp 326 - 345.

98. Snekkenes, E. (1991), "Exploring the BAN Approach to Protocol Analysis", *IEEE Symposium on Security and Privacy*, pp 171 - 181.

99. Snekkenes, E. (1992), "Roles in Cryptographic Protocols", *IEEE Symposium on Research and Security*, pp 105 - 119.

100. Meadows, C (1991), "A System for the Specification and Analysis of Key Management Protocols", *IEEE Symposium on Research and Security*, pp 182 - 195.

101. Glasgow, J. MacEwen, G. and Panangaden, P (1992), "A Logic for Reasoning About Security", *ACM Transactions on Computer Systems*, v10, n3, pp 226 - 264.

102. Rueppel, R.A. (1991), "A Formal Approach to Security Architectures", *Advances in Cryptography: Eurocrypt 91*, Springer-Verlag, pp 387 - 397.

103. Diffie, W. and Hellman, M.E. (1976), "Multiuser Cryptographic Techniques", *Proceedings of the National Computer Conference*, pp 109 - 112.

104. Knapskog, S.J. (1990), "Formal Specification and Verification of Secure Communication Protocols", *Lecture Notes in Computing Science 1990*, v453, pp 58 - 73.

105. Information Processing Systems - Open Systems Interconnection - Specification of Abstract Syntax Notation (ASN.1), CCITT Recommendation X.208, *Blue Book*, 1988.

106. Varadharajan, V (1990), "Petri Net Based Modelling of Information Flow Security Requirements, *Proceedings of the Computer Security Foundations Workshop III*, IEEE, pp 51 - 61.

107. Yahalom, R., Klein, B, and Beth, T. (1993), "Trust Relationships in Secure Systems - A Distributed Authentication Perspective", *IEEE Computer Society Symposium on Research in Security and Privacy*, ch. 17, pp 150 - 164.

108. Bird, R., Gopal, I, Herzberg, A., Janson, P., Kutten, S., Molva, R. and Yung, M. (1991), "Systematic Design of Two-Party Authentication Protocols", *Advances in Cryptology: Crypto 91*, Lecture Notes in Computer Science, Springer-Verlag, pp 44 - 61.

109. Cooke, J.C. and Brewster, R.L. (1992), "Security in Personal Communication Systems", *Fourth Bangor Symposium on Communications*, IEE, pp 172 - 175.

110. Vary, P. (1989), "Implementation Aspects of the Pan-European Digital Mobile Radio System", *Compeuro 89*, IEEE, pp 4.17 - 4.22.

111. Cooke, J.C. and Brewster, R.L. (1992), "Cryptographic Security Techniques for Digital Mobile Telephones - Part II", *IEEE International Conference on Selected Topics in Wireless Communications*, pp 425 - 428.

112. Cooke, J.C. and Brewster, R.L. (1992), "Cryptographic Security Techniques for Digital Mobile Telephones - Part I", *International Conference on Private Switching Systems and Networks*, IEE, pp 123 - 130.

113. ETSI, (1990), "Mobile Radio Interface Layer 3 Specification", ETSI/TC GSM, Recommendation 04.08.

114. ETSI, (1991), "Organisation of Subscriber Data", ETSI/TC GSM, Recommendation 03.08.

115. Audestad, J.A. (1989), "Intelligence in Public Land Mobile Networks: Use of the Mobile Application Part", *International Conference on Intelligent Networks*, IEEE, pp 57 - 61.

116. ETSI, (1990), "Radio Sub-System Link Control", ETSI/TC GSM, Recommendation 05.08.

117. Hodges, M.R.L. (1990), "The GSM Radio Interface", *British Telecom Technical Journal*, v8, n1, pp 31 - 43.

118. Rast, H.R., Wolfenstetter, K.-D. and Mouley, M. (1987), "Security Mechanisms in the Future European Digital Cellular System", *International Conference on Digital Land Mobile Communications*, IEEE, pp 124 - 133.

119. ETSI, (1991), "Security Related Network Functions", ETSI/TC GSM, Recommendation 03.20.

120. ETSI, (1991), "Digital European Cordless Telecommunications Common Interface", pt. 7: Security Features, prETS 300 175 - 7.

121. Walker, M. (1992), "Security in Mobile and Cordless Telecommunications", *International Computer Conference in Europe: Computer Systems and Software Engineering (Compeuro 92)*, IEEE, pp 493 - 496.

122. Cooke, J.C. and Brewster, R.L. (1993), "The Use of Smart Cards in Personal Communication Systems Security", *IEE Conference on Telecommunications*, pp 246 - 251.

123. Peyret, P., Lisimaque, G. and Chua, T.Y. (1990), "Smart Cards Provide Very High Security and Flexibility in Subscribers Management", *IEEE Transactions on Consumer Electronics*, v36, n3, pp 744 - 752.

124. ETSI, (1992), "Specifications of the SIM-ME Interface", ETSI/TC GSM, Recommendation 11.11.

125. Cooke, J.C. and Brewster, R.L. (1994), "Cryptographic Algorithms and Protocols for Personal Communication Systems Security", *IEE Colloquium on Security and Cryptography Applications to Radio Systems*, pp 8.1 - 8.6.

126. ETSI, (1993), "Security Principles for the Universal Mobile Telecommunications System (UMTS)", Draft ETR/SMG - 50901 (UMTS 09 - 01), version 0.1.5.

127. ITU, (1993), "Working Document Towards the New Recommendation: Security Principles for FPLMTS (FPLMTS:SCRT)", version 3.

128. ITU, (1993), "Working Document Towards the New Recommendation: Requirements on Security Procedures for FPLMTS (FPLMTS:SECPR)", version 3.

129. Vodafone Ltd., GPT Ltd. and Royal Holloway (University of London) (1994), "Security Features for Third Generation Systems", version 1.

APPENDIX

A listing of the *C* source code for the suite of programs used to model the search for large prime numbers on a Sun™ *Sparc10* Workstation.

```
/**********************************************************************/
/* LargeOp.h: header of LargeOp.c                                      */
/*                                                                     */
/**********************************************************************/


#define OPT_MUL

#define MAX_DIGITS          100
typedef unsigned char       BYTE;
typedef struct
                    {
                            int      Size;
                            BYTE  Digits[MAX_DIGITS];
                    }LARGE_INT;

LARGE_INT  One;
LARGE_INT  Two;
LARGE_INT  Three;


extern  void    InitLarge ();   /* Initialise tools for operation with LARGE_INT */
extern  void    Reset();        /* Set Op to zero */
extern  void    LargeCopy (); /* Copy Op1 in Op2 */
extern  void    ShiftByteLeft ();       /* Shift Op leftward of n bytes */
extern  void    ShiftByteRight ();      /* Shift Op rightward of n bytes */
extern  char    Comp();         /* return sign of Op1-Op2 (ie -1, 0 or 1) */
extern  void    Add();          /* OpT=Op1+Op2 */
extern  short   Sub ();         /* OpT=Op1+Op2 */
extern  void    Mul ();         /* OpT=Op1*Op2 */
extern  void    Div ();         /* Quotient=Op1 DIV Op2 with Remainder */
extern  void    Display ();     /* Display Op */
extern  void    GenerateRandom ();  /* Generate a random LARGE_INT */
extern  void    PowerMod();  /* OpT=(Op to the Power)(mod Mod) */
extern  unsigned        LargeToUns();           /* OpL->OpU */
extern  void    UnsToLarge();           /* OpU->OpL */
extern  void    LargeSQRT();            /* Res#SQRT(Op), Op=Res^2+Rem */
```

```
/****************************************************************************/
/*                    MODULE LargeOp.c                                      */
/*  Library of functions used to operate LARGE_INTs : Add, Sub, Mul, Div,   */
/*  Reset, ShiftByteLeft, ShiftByteRight, LargeCopy, Display,Comp,          */
/*  GenerateRandom.                                                         */
/****************************************************************************/


#include "LargeOp.h"
#include <memory.h>
#include <sys/time.h>
#include <stdio.h>


BYTE          Bits[8];        /* Used for Multilpication (Mul) */

#ifdef OPT_MUL      /* This is used in case of optimized Mul */
unsigned short      MulArray[256][256];
#endif



/****************************************************************************/
/* InitLarge : Initialises values and arrays used by the LARGE_INT functions.  */
/*    It MUST be launched before any operation on LARGE_INTs is used.          */
/****************************************************************************/

void    InitLarge ()     /* Initialise tools for operation with LARGE_INT */

{
int             i,j;
unsigned int    Prod;

        One.Size=1;             /* Initialisation of the LARGE_INTs One, Two & Three */
        One.Digits[0]=1;

        Two.Size=1;
        Two.Digits[0]=2;

        Three.Size=1;
        Three.Digits[0]=3;

        Bits[0]=1;
        for (i=1 ; i<=7 ; i++)
                Bits[i]=Bits[i-1]<<1;
 #ifdef OPT_MUL
        for (i=0 ; i<=255 ; i++)
                for (j=0 ; j<=256 ; j++)
                        MulArray[i][j] = (unsigned short)(i*j);
 #endif

 }
```

```
/****************************************************************************/
/*  Reset : Sets a LARGE_INT to ZERO                                        */
/*                                                                          */
/****************************************************************************/

void    Reset (Op)  /* Set Op to zero */
LARGE_INT *Op;

{
        memset(Op, 0, sizeof(LARGE_INT));
}


/****************************************************************************/
/*  LargeCopy : Copies the value of a LARGE_INT in another                  */
/*                                                                          */
/****************************************************************************/

void    LargeCopy (Op1, Op2)  /* Copies Op1 in Op2 */
LARGE_INT *Op1, *Op2;

{
        memcpy (Op2, Op1, Op1->Size+sizeof(int));
}


/****************************************************************************/
/*  ShiftByteLeft : Shifts up bytes of a LARGE_INT                          */
/*                                                                          */
/****************************************************************************/

void    ShiftByteLeft (Op, n)  /* Shift Op leftward of n bytes */
LARGE_INT  *Op;
int             n;

{
int     s,t;

        if (Op->Size && n)
        {
                for (s=Op->Size-1, t=s+n ; s>=0 ; s--, t--)
                        Op->Digits[t]=Op->Digits[s];
                for ( ; t>=0 ; t--)
                        Op->Digits[t]=0;
                Op->Size=Op->Size+n;
        }
}


/****************************************************************************/
/*  ShiftByteRight : Shifts down bytes of a LARGE_INT                       */
/*                                                                          */
/****************************************************************************/

void    ShiftByteRight (Op, n)          /* Shift Op rightward of n bytes */

LARGE_INT  *Op;
int             n;
```

```
{
int     s,t;
        if(n>=Op->Size)
        {
                Op->Size=0;
                return;
        }
        if (Op->Size && n)
        {
                for (t=0, s=n ; s<=Op->Size-1 ; s++,t++)
                        Op->Digits[t]=Op->Digits[s];
                Op->Size=Op->Size-n;
        }
}


/**************************************************************************/
/* ShiftBitLeft : Shifts up bits of a LARGE_INT                           */
/*                                                                        */
/**************************************************************************/

void    ShiftBitLeft (Op, n)    /* Shift Op leftward of n bits */
LARGE_INT  *Op;
int             n;

{
int     i,nn;
BYTE  CurByte;

        if (Op->Size && n)
        {
                if(n>7)
                        ShiftByteLeft(Op,(int)(n/8));
                nn=n-8*(int)(n/8);
                if(nn)
                {
                        Op->Digits[Op->Size]=0;
                        for (i=Op->Size ; i>0 ; i--)
                        {
                                Op->Digits[i]<<=nn;
                                CurByte=Op->Digits[i-1]>>(8-nn);
                                Op->Digits[i]|=CurByte;
                        }
                        Op->Digits[0]<<=nn;
                        if(Op->Digits[Op->Size])
                                Op->Size++;
                }
        }
}


/**************************************************************************/
/* ShiftBitRight : Shifts down bits of a LARGE_INT                        */
/*                                                                        */
/**************************************************************************/

void    ShiftBitRight (Op, n)           /* Shift Op rightward of n bits */
```

```
LARGE_INT  *Op;
int             n;

{
int      i,nn;
BYTE  CurByte;

        if (Op->Size && n)
        {
                if(n>7)
                        ShiftByteRight(Op,(int)(n/8));
                nn=n-8*(int)(n/8);
                if(nn && Op->Size)
                {
                        for (i=0 ; i<Op->Size-1 ; i++)
                        {
                                Op->Digits[i]>>=nn;
                                CurByte=Op->Digits[i+1]<<(8-nn);
                                Op->Digits[i]|=CurByte;
                        }
                        Op->Digits[i]>>=nn;
                        if(!Op->Digits[i])
                                Op->Size--;
                }
        }
}



/****************************************************************************/
/* Comp : Compares two LARGE_INTs and returns the sign of the difference      */
/*                                                                          */
/****************************************************************************/

char    Comp(Op1, Op2)        /* return sign of Op1-Op2 (ie -1, 0 or 1) */

LARGE_INT *Op1, *Op2;

{
int       DiffSize, DiffVal;
int       i;

/* In order to optimize the process, sizes are compared first */
        DiffSize = Op1->Size-Op2->Size;
        if (DiffSize > 0)
                return  1;
            else
                if (DiffSize < 0)
                        return -1;
                    else
                        {
/* If sizes are equal then bytes of the same rank are compared starting with the most significant
ones */
                                i = Op1->Size-1;
                                do
                                {
                                        DiffVal =
                                        Op1->Digits[i]-Op2->Digits[i--];
                                        if (DiffVal > 0)
                                                return 1;
                                        if (DiffVal < 0)
                                                return -1;
                                } while (i>=0);
                                return 0;
                        }
}


/*************************************************************************/
/*  Add : Performs the addition of two LARGE_INTs                        */
/*                                                                       */
/*************************************************************************/

void    Add(Op1, Op2, OpT)          /* OpT=Op1+Op2 */

LARGE_INT *Op1, *Op2, *OpT;

{
int             i;
unsigned        Inter, Carry;
LARGE_INT   Buffer;


        if(!Op1->Size || !Op2->Size)
        {
                if(!Op1->Size)
                        LargeCopy(Op2,OpT);
                else
                        LargeCopy(Op1,OpT);
                return;
        }
        Carry = 0;
        i = 0;
        if(Op1->Size >= Op2->Size)
        {
                do
                {
                        Inter=Op1->Digits[i]+Op2->Digits[i]+Carry;
                        Buffer.Digits[i++] =(BYTE)(Inter & 0xFF);
                        Carry = (BYTE)(Inter>>8);
                }while(i<Op2->Size);
                for( ; i<Op1->Size ; i++)
                {
                        Inter=Op1->Digits[i]+Carry;
                        Buffer.Digits[i]=(BYTE)(Inter & 0xFF);
                        Carry = (BYTE)(Inter>>8);
                }
        }
        else
        {
                do
                {
                        Inter=Op1->Digits[i]+Op2->Digits[i]+Carry;
                        Buffer.Digits[i++] =(BYTE)(Inter & 0xFF);
                        Carry = (BYTE)(Inter>>8);
                }while(i<Op1->Size);
```

```
          for( ; i<Op2->Size ; i++)
          {
                    Inter=Op2->Digits[i]+Carry;
                    Buffer.Digits[i]=(BYTE)(Inter & 0xFF);
                    Carry = (BYTE)(Inter>>8);
          }
     }
     Buffer.Size = i;
     if (Carry)
     {
          Buffer.Digits[i] = (BYTE)Carry;
          Buffer.Size++;
     }
     LargeCopy(&Buffer, OpT);
}



/************************************************************************/
/* Sub : Performs the substraction of two LARGE_INTs                    */
/* If the result would be negative, this function doesn't perform the   */
/* substraction and returns -1. Otherwise it returns 1                  */
/************************************************************************/

short   Sub (Op1, Op2, OpT)          /* OpT=Op1-Op2 */

LARGE_INT *Op1, *Op2, *OpT;

{
int            i, CurrentSize;
long           OpSub;
LARGE_INT      Buffer;
short          Test;

     if (!Op2->Size)          /* Does the substracted operand equal 0 ? */
     {
          LargeCopy(Op1, OpT);          /* Then no computation required */
          return 1;
     }
     switch(Comp(Op1, Op2))       /* Result is 0 since operands are equal */
     {
          case 0:
               OpT->Size = 0;
               return 1;
          case -1:
               return -1;
          /* No negative numbers in the LARGE_INT set */
          case 1:
               CurrentSize = 0;
               i=0;
               OpSub = 0;
               do
               {
                    OpSub = OpSub+Op2->Digits[i];
                    if (Op1->Digits[i] < OpSub)
                    {
/* In this case, 0x100 has to be added to the Op1 digit to perform the sub */
                              Buffer.Digits[i]=
                                        (BYTE)(0x0100+Op1->Digits[i]-OpSub);
                                        OpSub = 1; /* The carry */
                                }
                                else
                                {
                                        Buffer.Digits[i]=
                                         (BYTE)(Op1->Digits[i]-OpSub);
                                        OpSub = 0;
                                }
                                if (Buffer.Digits[i])
                                        CurrentSize = i+1;
                        } while(++i < Op2->Size);
                        for( ; i<Op1->Size ; i++)
                                {
                                        if (Op1->Digits[i] < OpSub)
                                        {
                                                Buffer.Digits[i]=0xFF;
                                                OpSub = 1;      /* The carry */
                                        }
                                        else
                                        {
                                                Buffer.Digits[i]=
                                                 Op1->Digits[i]-OpSub;
                                                OpSub = 0;
                                        }
                                        if (Buffer.Digits[i])
                                                CurrentSize = i+1;
                                }
                        Buffer.Size = CurrentSize;
                        LargeCopy(&Buffer, OpT);
                        return 1;
        }
}


/**************************************************************************/
/*  Mul : Performs the multiplication of two LARGE_INTs                   */
/*                                                                        */
/**************************************************************************/

void    Mul (Op1, Op2, OpT)             /* OpT=Op1*Op2 */

LARGE_INT *Op1, *Op2, *OpT;

{
#ifndef OPT_MUL
LARGE_INT  Buffer, Buffer2;
int             i1, i2;
unsigned int    Product;

{
        if(!Op1->Size || !Op2->Size)
        {
                Reset(OpT);
                return;
        }
        else
        {
```

```
			Reset (&Buffer2);
			for (i2=0 ; i2<Op2->Size ; i2++)
			{
				Reset(&Buffer);
				for (i1=0 ; i1<Op1->Size ; i1++)
				{
					Product=(unsignedint)
					 (Op1->Digits[i1]*Op2->Digits[i2]
					 +Buffer.Digits[i1+i2]);
					Buffer.Digits[i1+i2]=(BYTE)(Product&0x00FF);
					Buffer.Digits[i1+i2+1]=
					 (BYTE)((Product>>8)&0x00FF);
				}
				Buffer.Size=i1+i2;
				if(Buffer.Digits[i1+i2])
					Buffer.Size++;
				Add(&Buffer, &Buffer2, &Buffer2);
			}
		LargeCopy (&Buffer2, OpT);
		}
}

#endif

#ifdef OPT_MUL
LARGE_INT  Buffer, Buffer2;
int             i1, i2, i,j;
unsigned short          Carry,Next;

{
		if(!Op1->Size || !Op2->Size)
		{
			Reset(OpT);
			return;
		}
		else
		{
			Reset (&Buffer2);
			for (i2=0 ; i2<Op2->Size ; i2++)
			{
				Reset(&Buffer);
				Carry=0;
				for (i1=0 ; i1<Op1->Size ; i1++)
				{
					Next=MulArray[(int)(Op1->Digits[i1])]
					[(int)(Op2->Digits[i2])]+Carry;
					Buffer.Digits[i1+i2]=(BYTE)(Next&0xFF);
					Carry=Next>>8;
				}
				Buffer.Size=i1+i2;
				if(Carry)
				{
					Buffer.Digits[i1+i2]=(BYTE)(Carry);
					Buffer.Size++;
				}
				Add(&Buffer, &Buffer2, &Buffer2);
			}
		LargeCopy (&Buffer2, OpT);
		}
```

```
}
#endif
}


/****************************************************************************/
/*  Div : Performs the division of two LARGE_INTs                           */
/*                                                                          */
/****************************************************************************/

void    Div (Op1, Op2, Quotient, Remainder)
                    /* Quotient=Op1 DIV Op2 with Remainder */
LARGE_INT  *Op1, *Op2, *Quotient, *Remainder;

{
int             i,j, Flag, BitShiftCount, ByteShiftCount;
LARGE_INT  Op1Copy, Op2Copy, Quo, Bin;

        if(Comp(Op2,&One)==0)
        {
                LargeCopy(Op1,Quotient);
                Reset(Remainder);
                return;
        }
        if(!Op1->Size)
        {
                Reset(Quotient);
                Reset(Remainder);
                return;
        }
        switch (Comp(Op1,Op2))
        {
        /* These cases are processed here to save processor time */
                case    -1 :    LargeCopy(Op1,Remainder);
                                Reset(Quotient);
                                return;
                case    0 :     Quotient->Size=1;
                                Quotient->Digits[0]=1;
                                Reset(Remainder);
                                return;
                default   :
                Reset(&Quo);
                Flag=0;
                BitShiftCount=0;
                LargeCopy(Op1,&Op1Copy);
                LargeCopy(Op2,&Op2Copy);
                ByteShiftCount=Op1Copy.Size-Op2Copy.Size;
                ShiftByteLeft(&Op2Copy,ByteShiftCount);
                while(Comp(&Op2Copy,&Op1Copy)==-1)
                {
                        ShiftBitLeft(&Op2Copy,1);
                        BitShiftCount++;
                }
                while(Comp(&Op2Copy,&Op1Copy)==1)
                {
                        ShiftBitRight(&Op2Copy,1);
                        BitShiftCount--;
                }
                BitShiftCount+=(8*ByteShiftCount);
```

```
            do
            { /*
                    printf("Op1Copy=");
                    Display(&Op1Copy);
                    printf("Op2Copy=");
                    Display(&Op2Copy); */
                    if(Comp(&Op1Copy,&Op2Copy)!=-1)
                    {
                            Quo.Digits[0]|=0x01;
                            Sub(&Op1Copy,&Op2Copy,&Op1Copy);
                            if(!Flag)
                            {
                                    Quo.Size=1;
                                    Flag=1;
                            }
                    }
                    if(Comp(&Op1Copy, Op2)==-1)
                    {
                            ShiftBitLeft(&Quo,BitShiftCount);
                            LargeCopy(&Op1Copy,Remainder);
                            LargeCopy(&Quo,Quotient);
                            return;
                    }
                    ShiftBitRight(&Op2Copy,1);
                    BitShiftCount--;
                    ShiftBitLeft(&Quo, 1);
            } while(1);
        }
}


/**************************************************************************/
/* Display : Displays a LARGE_INT on the screen                           */
/*                                                                        */
/**************************************************************************/

void    Display (Op)            /* Display Op */

LARGE_INT  *Op;

{
int     i;

        if(!Op->Size)
                printf(" 00");
        else
                for (i=Op->Size-1 ; i>=0 ; i--)
                        printf (" %02X",Op->Digits[i]);
        printf ("\n");
}


/**************************************************************************/
/* GenerateRandom : Generate a random LARGE_INT of a size determined by   */
/* Min and Max                                                            */
/**************************************************************************/

void    GenerateRandom (Op, Min, Max)     /* Generate a random LARGE_INT */
```

```
LARGE_INT  *Op;
int             Min,   /* Minimum number of digits required */
                Max;   /* Maximum number of digits required */

{
int     i, Flag;
struct  timeval                 tp;
struct  timezone        tzp;

        Flag=0;
        for (i=0 ; i<Max ; i++)
        {
                do
                {
                        gettimeofday(&tp,&tzp);
                        Op->Digits[i]=(BYTE)(tp.tv_usec & 0x00FF);
                        gettimeofday(&tp,&tzp);
                } while (tp.tv_usec & 0x0081);
                if (!Flag && (i>=Min-1) && Op->Digits[i])
                        Flag=1;
                while ((i==Max-1) && !Flag && !Op->Digits[i])
                {
                        gettimeofday(&tp,&tzp);
                        Op->Digits[i]=(BYTE)(tp.tv_usec & 0x00FF);
                }
                if (Op->Digits[i])
                        Op->Size=i+1;
        }
}


/**************************************************************************/
/*  PowerMod : Compute the power of a number modulo another number        */
/*                                                                        */
/**************************************************************************/

void    PowerMod(Op, Power, Mod, OpT)   /* OpT=(Op to the Power)(mod Mod) */
LARGE_INT  *Op, *Power, *Mod, *OpT;

{
int             i,j;
BYTE            Mask;
LARGE_INT  CurrentPower,Quotient;

        LargeCopy(&One, OpT);
        LargeCopy(Op, &CurrentPower);
        for(i=0 ; i<Power->Size ; i++)
        {
                Mask=1;
                for(j=0 ; j<=7 ; j++)
                {
                        if(Mask & Power->Digits[i])
                        {
                                Mul(OpT, &CurrentPower, OpT);
                                Div(OpT, Mod, &Quotient, OpT);
                        }
                        Mask<<=1;
                        Mul(&CurrentPower, &CurrentPower, &CurrentPower);
                        Div(&CurrentPower, Mod, &Quotient, &CurrentPower);
                }
        }
}

/****************************************************************************/
/* LargeToUns : Translate a large integer into an unsigned                  */
/*                                                                          */
/****************************************************************************/

unsigned        LargeToUns(OpL)     /* OpL->OpU */
LARGE_INT  *OpL;

{
unsigned        Res;
int             i;

        if(OpL->Size)
        {
                Res=0;
                for(i=0 ; i<OpL->Size ; i++)
                        Res+=(unsigned)(OpL->Digits[i]<<(i*8));
                return Res;
        }
                return 0;
}


/****************************************************************************/
/* UnsToLarge : Translate an unsigned into a large integer                  */
/*                                                                          */
/****************************************************************************/

void    UnsToLarge(OpU,OpL)        /* OpU->OpL */
LARGE_INT  *OpL;
unsigned        OpU;

{
unsigned        Res;

        OpL->Digits[0]=(BYTE)(OpU&0xFF);
        OpL->Digits[1]=(BYTE)((OpU>>8)&0xFF);
        OpL->Digits[2]=(BYTE)((OpU>>16)&0xFF);
        OpL->Digits[3]=(BYTE)((OpU>>24)&0xFF);
        if(OpU<=0xFF)
                OpL->Size=1;
        else
                if(OpU<=0xFFFF)
                        OpL->Size=2;
                else
                        if(OpU<=0xFFFFFF)
                                OpL->Size=3;
                        else
                                OpL->Size=4;

}
```

```
/**************************************************************************/
/*  LargeSQRT : Finds the square root of a Large integer                  */
/*                                                                        */
/**************************************************************************/

void    LargeSQRT(Op, Res, Rem)   /* Res#SQRT(Op), Op=Res^2+Rem */
LARGE_INT  *Op, *Res, *Rem;

{
LARGE_INT  Current, r, Bin;

        LargeCopy(Op, &Current);
        do
        {
                LargeCopy(&Current, &r);
                Div(Op, &r, &Current, &Bin);
                Display(&Current);
                Add(&Current, &r, &Current);
                ShiftBitRight(&Current, 1);
                Display(&r);
        }while(Comp(&r,&Current)!=0);
        LargeCopy(&r, Res);
        Mul(&r,&r,&Current);
        Sub(Op, &Current, Rem);
}
```

```
/****************************************************************************/
/*  PrimeTools.h: header of PrimeTools.c                                    */
/*                                                                          */
/****************************************************************************/


#define         SP_NUMBER           1000
#define         SOLOVAY_ITER        20
extern  int     IsGCD1();     /*    Returns 1 if the Greatest Common Divisor of Val1 and
                              Val2 is 1, otherwise returns 0 */
extern  int     Jacobi();
extern  void    LargePrimeHunt();
extern  int     MakeSurePrime();
```

```
/*************************************************************************/
/*  PrimeTools.c        This program comtains the functions useful to    */
/*  searching for large prime numbers                                     */
/*************************************************************************/



#include "LargeOp.h"
#include "PrimeTools.h"
#include <memory.h>
#include <stdio.h>


typedef         struct
                        {
                        unsigned        Value, Counter;
                        }S_PRIME;

S_PRIME         SmallPrime[SP_NUMBER];


int     IsGCD1(Val1, Val2)              /*      Returns 1 if the Greatest Common Divisor of
                                                Val1 and Val2 is 1, otherwise returns 0 */
LARGE_INT  *Val1, *Val2;

{
LARGE_INT  Quotient, Remainder;
int             Yes;

        Yes=0;
        Div(Val1, Val2, &Quotient, &Remainder);
        if(Comp(&Remainder, &One)==0)
                return 1;
          else
                if(Remainder.Size==0)
                        return 0;
                else
                        Yes=IsGCD1(Val2, &Remainder);
        if(Yes)
                return 1;
          else
                return 0;
}


void    LargePrimeHunt(MinSize, MaxSize, PrimeFound)

int             MinSize, MaxSize;
LARGE_INT  *PrimeFound;

{
LARGE_INT  Seed, Rem, Prime, Buffer, Buffer2, RefuseBin, A;
int             n,i,j,FlagGotOne, FlagComposite, JacobiComp;
FILE            *fp;
unsigned        LSB, Result;

        printf("Reading file...\n");
        fp = fopen("SPrimesList","r");
        for(i=0 ; i<=SP_NUMBER-1 ; i++)
```

```
{
        n=fread(&SmallPrime[i].Value,sizeof(unsigned),1,fp);
        SmallPrime[i].Counter=0;
}
fclose(fp);
printf("Done!\n");
FlagGotOne=0;
GenerateRandom(&Seed,MinSize, MaxSize);
Seed.Digits[0]|=0x01; /*Always odd*/
printf("Initializing SmallPrime array...\n");
for(i=0 ; i<=SP_NUMBER-1 ; i++)
{
        UnsToLarge(SmallPrime[i].Value, &Prime);
        Div(&Seed, &Prime, &RefuseBin, &Rem);
        if(Rem.Digits[0]&0x01)
        {
                ShiftBitRight(&Prime, 1);
                Sub(&Rem, &One, &Rem);
        }
        ShiftBitRight(&Rem,1);
        Sub(&Prime, &Rem, &RefuseBin);
        SmallPrime[i].Counter=LargeToUns(&RefuseBin);
}
printf("Done!\n");
do
{
        Display(&Seed);
        FlagComposite=0;
        for(i=0 ; i<=SP_NUMBER-1 ; i++)
        {
                if(!SmallPrime[i].Counter)
                {
                        SmallPrime[i].Counter=SmallPrime[i].Value;
                        FlagComposite=1;
                }
                SmallPrime[i].Counter--;
        }
        for(j=0 ; j<SOLOVAY_ITER && !FlagComposite ; j++)
        {
                printf("*");
                GenerateRandom(&A, (int)(MinSize-1),(int)(MinSize-1));
                if(IsGCD1(&A,&Seed))
                {
                        Sub(&Seed, &One, &Buffer);
                        ShiftBitRight(&Buffer,1);
                        PowerMod(&A, &Buffer, &Seed, &Buffer);
                        switch(Comp(&Buffer, &One))
                        {
                        case 0 :        JacobiComp=1;
                                        break;
                        case 1 :        Add(&Buffer,&One,&Buffer2);
                                        if(!Comp(&Buffer2,&Seed))
                                                JacobiComp=-1;
                                        break;
                        default:        JacobiComp=0;
                        }
                        if(JacobiComp)
                                if(Jacobi(&A, &Seed)!=JacobiComp)
                                        FlagComposite=1;
```

```
                    }
                    else
                            FlagComposite=1;
            }
            printf("\n");
            if(!FlagComposite)
                    exit(1); /*
                    if(FlagGotOne=MakeSurePrime(&Seed))
                            LargeCopy(&Seed, PrimeFound); */
            Add(&Seed, &Two, &Seed);
    }while(!FlagGotOne);
}


int     Jacobi(a, n)
LARGE_INT  *a, *n;

{
LARGE_INT  nModa, RefuseBin;
int     Sign,Inter;

        Display(a);
        Display(n);
        if(!Comp(a,&One))
                return 1;
        if(a->Digits[0]&0x01)
        {
                Sign=(n->Digits[0]*n->Digits[0]-1)&0x08;
                ShiftBitRight(a,1);
                Inter=Jacobi(a,n);
                if(Sign)
                        return -Inter;
                else
                        return Inter;
        }
        else
        {
                Sign=((n->Digits[0]-1)*(a->Digits[0]-1))&0x04;
                Div(n, a, &nModa, &RefuseBin);
                Inter=Jacobi(&nModa,a);
                if(Sign)
                        return -Inter;
                else
                        return Inter;
        }
}

int     MakeSurePrime(Val)

LARGE_INT  *Val;

{
int             FlagComp;
LARGE_INT  i,Rem,Quo,MaxDiv;

        LargeCopy(Val,&MaxDiv);
        ShiftBitRight(&MaxDiv,2);
        Mul(&MaxDiv,&Three,&MaxDiv);
        printf("Checking for primalty of ");
        Display(Val);
        UnsToLarge(SmallPrime[SP_NUMBER-1].Value+2,&i);
        FlagComp=0;
        do
        {
                Div(Val,&i,&Quo,&Rem);
                if(Rem.Size==0)
                        FlagComp=1;
                   else
                        Add(&i,&Two,&i);
        }while(!FlagComp && (Comp(&i,&MaxDiv)==-1));
        if(!FlagComp)
        {
                printf("Prime number !!\n");
                return 1;
        }
        else
        {
                printf("Sorry it is composite.\n");
                return 0;
        }

}
```

```
/*****************************************************************************/
/* PrimeHunter.c : This program searches for prime numbers from 3 to NbPrimes  */
/* and stores them in the file SPrimesList                                      */
/*****************************************************************************/

#include <stdio.h>
#include <math.h>

main ()

{
FILE    *fp;
unsigned int    NbPrimes, CurVal, i;
double                  MaxDiv, Val;
int             n,FlagComp;

        printf("How many ? ");
        scanf("%d",&NbPrimes);
        fp = fopen("SPrimesList","w");
        CurVal=3;
        n=fwrite(&CurVal,sizeof(unsigned int),1,fp);
        NbPrimes--;
        while(NbPrimes)
        {
                CurVal+=2;
                MaxDiv=sqrt((double)CurVal);
                i=3;
                FlagComp=0;
                do
                {
                        Val=CurVal/i;
                        if(i*Val==CurVal)
                                FlagComp=1;
                           else
                                i+=2;
                }while(!FlagComp && i<=MaxDiv);
                if(!FlagComp)
                {
                        n=fwrite(&CurVal,sizeof(unsigned int),1,fp);
                        NbPrimes--;
                }
        }
        fclose(fp);
 }
```

```
/**************************************************************************/
/* ProbEval.c : This program computes the probability of a number of being       */
/* composite by considering the multiples eliminated                             */
/**************************************************************************/

#include <stdio.h>

main(argc,argv)         /* this parameter is the initial probability */
int     argc;
char    *argv[];

{
FILE            *fp,*fp2;
char            ch;
unsigned        Val;
int             i,n;
float           Prob;

        n=sscanf(argv[1],"%e",&Prob);
        printf("Go...%d  %g\n",n,Prob);
        i=0;
        fp2 = fopen("ProbList","w");
        fp = fopen("SPrimesList","r");
        for(i=1 ; i<=20000 ; i++)
        {
                n=fread(&Val,sizeof(unsigned),1,fp);
                Prob=(float)(Prob*(Val-1)/Val);
                fprintf(fp2,"No %d,  Value:%d,  Prob:%g\n",i,Val,Prob);
        }
        fclose(fp);
        fclose(fp2);
        return;
 }
```