p-ADIC NUMBER THEORY AND ITS APPLICATIONS IN A CRYPTOGRAPHIC FORM

by

RAOUF N. GORGUI-NAGUIB

A Thesis Submitted for the Degree of
Doctor of Philosophy
of the University of London

Department of Electrical Engineering
Imperial College of Science and Technology
University of London

June 1986

ABSTRACT

This thesis is concerned with a study of p-adic number theory and its application in developing a secure public-key cryptographic system.

p-adic number systems may have either an infinite or finite structure. First, the infinite p-adic expansion is considered with particular emphasis on the finite representation of such infinite sequences. Proof of periodicity and an algorithm for its computation are given. An algorithm for the conversion from these variable-length representations to the field of rational numbers is also developed based on existing algorithms for the conversion in the case of finite p-adic systems.

However, by considering the arithmetic in segmented p-adic fields based on Hensel codes, it is shown that previously derived algorithms tend to give erroneous results. These limitations are corrected and modified algorithms for closed finite p-adic arithmetic operations are presented.

Due to their inherent structure, finite p-adic number systems do not lend themselves to a secure implementation of a p-adic-based cryptosystem. Consequently, a new public-key encryption system based on variable-length p-adic number structures is proposed. The system has the property of combining the advantages of both the Rivest-Shamir-Adleman (RSA) and Diffie-Hellman algorithms and of exploiting the pseudo-randomness of the p-adic numbers in the p-adic field Q_p to generate the ciphertext.

This first scheme is then extended to the g-adic ring, Q_g, where the ciphertext, now, consists of g-adic numbers. The decryption algorithm first converts the g-adic sequence into p-adic numbers using information sent by the source and based and discrete logarithms. Following that, a conversion back to the field of rational numbers enables the recipient to decipher the message.

Cryptanalytic approaches to break the two proposed schemes are then considered and short and long messages are simulated, first on a random basis, then based on the relative frequencies of the alphabetic characters in the English language and the distribution of p-adic numbers for different primes is studied.

Authentication in this system is discussed and a new digital signature procedure is introduced. Finally, an overall comparison between the system and existing public-key cryptosystems is performed.

ACKNOWLEDGEMENTS

The work reported in this thesis was supervised by Dr. Robert King. I am grateful for the many advices he gave and the critical discussions we had in the course of the preparation of the various papers and the thesis. I would also like to express my appreciation for creating a free atmosphere for research which contributed significantly to this work.

I gratefully acknowledge the financial supports given by the Committee of Vice-Chancellors and Principals (CVCP) of the Universities of the United Kingdom through their Overseas Research Students (ORS) award, and the University of London through the Sir Edward Stern Studentship award.

My special thanks to Dr. I. Habbab for the many stimulating discussions we had during the last few years and for the valuable advices he gave me regarding this work.

I wish to extend my thanks and appreciation to my colleagues, and especially to the members of Dr. King's research group, who helped in any way in the formulation of this thesis. Their constructive criticisms and the many different attacks they directed to the proposed cryptographic system were very important in ascertaining its theoretical and practical security.

Finally, I dedicate this thesis to May and to my parents, especially my father, to whom I know it means a great deal. Without their love, support and confidence in me, this work could not have been achieved.

TABLE OF CONTENTS

		page
ABSTRACT		2
ACKNOWLEDGEMENTS		4
TABLE OF CONTENTS		5
LIST OF TABLES		10
LIST OF FIGURES		11
LIST OF SYMBOLS		13
CHAPTER 1	INTRODUCTION	
1.1	Historical Background of Number Theory	15
1.2	Introduction to p-adic Number Theory	17
1.3	Brief Introduction to Cryptography	18
1.4	Outline of the Thesis	20
CHAPTER 2	VARIABLE-LENGTH p-ADIC NUMBER SYSTEMS	
2.1	Infinite p-adic Expansions	23
2.2	Krishnamurthy's Algorithm for Conversion from Rational Number to Infinite p-adic Form	26
2.3	Period Detection	27
2.3.1	Proof of Periodicity in Infinite p-adic Number Systems	27
2.3.2	Algorithm for the Period Computation	30
2.4	Algorithm for Computing the Variable-Length p-adic Expansions	36

page

CHAPTER 3 FINITE-SEGMENT p-ADIC NUMBER SYSTEMS

3.1 Mathematical Introduction to Finite p-adic Number Systems 39

3.2 Structure of Hensel Codes 41

3.3 Arithmetic Operations in Segmented p-adic Fields 43

3.3.1 p-adic Addition in $\hat{Q}_p$ 44

3.3.2 p-adic Subtraction in $\hat{Q}_p$ 46

3.3.3 p-adic Multiplication in $\hat{Q}_p$ 47

3.3.4 p-adic Division in $\hat{Q}_p$ 47

3.3.5 Limitations of Krishnamurthy's Algorithms 49

3.3.5.1 Limitations of the Addition Algorithm 49

3.3.5.2 Modified Addition Algorithm 50

3.3.5.3 Limitations of the Multiplication Algorithm 52

3.3.5.4 Modified Multiplication Algorithm 53

3.4 Conversion of Hensel Codes and Variable-Length p-adic Expansions to Rational Form 57

3.4.1 Method of Congruences Based on p-adic Weights 57

3.4.2 Method of Look-Up Tables 60

3.4.3 Method of Successive Addition 60

page

3.5 Limitations of the Use of Segmented p-adic Number Systems 61

CHAPTER 4 MATHEMATICAL THEORY OF CRYPTOGRAPHIC SYSTEMS

4.1 Introduction to Cryptosystems 63

4.2 Authentication and Digital Signature 66

4.3 Complexity Theory and the Theory of NP-Completeness 68

4.4 Public-Key Cryptology 70

4.4.1 The Diffie-Hellman System 71

4.4.2 Comments on the Diffie-Hellman System 74

4.4.3 The Merkle-Hellman System 75

4.4.4 Comments on the Merkle-Hellman System 78

4.4.5 The Rivest-Shamir-Adleman (RSA) System 80

4.4.6 Comments on the RSA System 82

CHAPTER 5 A NEW PUBLIC-KEY CRYPTOGRAPHIC SCHEME BASED ON p-ADIC NUMBER SYSTEMS

5.1 Introduction to the Proposed System 87

5.2 p-adic Code Structure for Cryptographic Implementation 88

5.3 Choice of Alphabet 93

5.4 Realization of the Scheme Based on p-adic Fields 94

5.4.1 The p-adic Encryption Algorithm 94

5.4.2 The p-adic Decryption Algorithm 96

page

5.5 Realization of the Scheme Based on g-adic Rings 98

5.5.1 The g-adic Encryption Algorithm 98

5.5.2 The g-adic Decryption Algorithm: The Decomposition of Q_g into Q_p 99

CHAPTER 6 EVALUATION OF THE PROPOSED SYSTEM

6.1 Cryptanalytic Approaches to Breaking the p-adic Based System 102

6.2 Attempts at Breaking the g-adic Based System-Diffusion and Confusion 132

6.3 Authentication and a New Digital Signature Procedure Based on p-adic Number Systems 136

6.4 Comparison with Other Public-Key Encryption Systems 140

CHAPTER 7 CONCLUSIONS

7.1 Summary of Contributions 145

7.2 Suggestions for Further Research 152

REFERENCES 155

APPENDIX A Table of $H(p,r,\alpha)$ Codes for $p = 5$, $r = 4$ 160

APPENDIX B Table of Variable-Length p-adic Codes for $p = 5$ and $\gamma = 17$ 163

page

APPENDICES C1 & C2 Computer Programs

Appendix C1 Program for the Variable-Length p-adic/Rational Conversion 167

Appendix C2 Program for the Finite-Segment p-adic Conversion and Full Arithmetic Package 183

page

LIST OF TABLES

Table 2.1(A)	Comparison of period lengths with the Euler ϕ-function for various primes p and denominators d.	34
Table 2.1(B)	Comparison of period lengths with the Euler ϕ-function for various larger primes p and the same denominators d	35
Table 3.1	Krishnamurthy's algorithm for segmented p-adic multiplication in the case of $\alpha = 1/15$ and $\beta = 5/4$ for $p = 5$ and $r = 4$	55
Table 3.2	Modified algorithm for segmented p-adic multiplication in the case of $\alpha = 1/15$ and $\beta = 5/4$ for $p = 5$ and $r = 4$	56
Table 4.1	Successive iterations for breaking the RSA system for the case $e = 17$, $n = 2773$ and $C = 948$	85
Table 6.1	Relative frequencies of the English characters and space	113

page

LIST OF FIGURES

Fig. 4.1	Block diagram of a conventional cryptosystem	65
Fig. 4.2	Containment relationships between complexity classes P, NP and NP-complete	69
Fig. 4.3	Block diagram of a public-key cryptosystem	72
Fig. 4.4	Flow of information in the Merkle-Hellman cryptosystem	79
Fig. 4.5	Flow of information in the Rivest-Shamir-Adleman (RSA) cryptosystem	83
Fig. 5.1	Structure of the variable-length p-adic code based on the period λ	89
Fig. 5.2	Flow of information in the p-adic based cryptosystem	95
Fig. 6.1	Frequency of occurrence of p-adic numbers for random message with equiprobable character frequencies- $p_{AB} = 2909$, $M = 100$	105
Fig.6.2	Frequency of occurrence of p-adic numbers for random message with equiprobable character frequencies - $p_{AB} = 2909$, $M = 1000$	106
Fig. 6.3	Frequency of occurrence of p-adic numbers for random message with equiprobable character frequencies - $p_{AB} = 2909$, $M = 10000$	107
Fig. 6.4	Entropy curves for random messages with equiprobable character frequencies - $M = 100$	108
Fig. 6.5	Entropy curves for random messages with equiprobable character frequencies - $M = 1000$	109
Fig. 6.6	Entropy curves for random messages with equiprobable character frequencies - $M = 10000$	110
Fig. 6.7	Frequency of occurrence of p-adic numbers for random message with relative frequencies of English characters - $p_{AB} = 2909$, $M = 100$	114
Fig. 6.8	Frequency of occurrence of p-adic numbers for random message with relative frequencies of English characters - $p_{AB} = 2909$, $M = 1000$	115
Fig. 6.9	Frequency of occurrence of p-adic numbers for random message with relative frequencies of English characters - $p_{AB} = 2909$, $M = 10000$	116
Fig. 6.10	Entropy curves for random messages with relative frequencies of English characters - $M = 100$	117

page

Fig. 6.11	Entropy curves for random messages with relative frequencies of English characters - M = 1000	118
Fig. 6.12	Entropy curves for random messages with relative frequencies of English characters - M = 10000	119
Fig. 6.13	Frequency of occurrence of p-adic numbers for the message: Making Confusion Worse Confounded - p_{AB} = 2909	120
Fig. 6.14	Frequency of occurrence of p-adic numbers for the message: It Is a Long Road that Has No Turning - p_{AB}= 2909	121
Fig. 6.15	Frequency of occurrence of p-adic numbers for the message: Making Confusion Worse Confounded - p_{AB} = 4001	122
Fig. 6.16	Frequency of occurrence of p-adic numbers for the message: It is a Long Road that Has No Turning - p_{AB} = 4001	123
Fig. 6.17	Frequency of occurrence of p-adic numbers for the message: Making Confusion Worse Confounded - p_{AB} = 9967	124
Fig. 6.18	Frequency of occurrence of p-adic numbers for the message: It is a Long Road that Has No Turning - p_{AB} = 9967	125
Fig. 6.19	Frequency of occurrence of p-adic numbers and period parameters for the message: Its All Greek to Me - p_{AB} = 4001	128
Fig. 6.20	Frequency of occurrence of p-adic numbers and period parameters (Log. scale) for the message: Its All greek To Me - p_{AB} = 4001	129
Fig. 6.21	Frequency of occurrence of p-adic numbers and modified period parameters for the message: Its All greek to Me p = 7489, p_{AB} = 4001	133
Fig. 6.22	Frequency of occurrence of p-adic numbers and modified period parmeters (Log.scale) for the message: Its All Greek To Me - p = 7489, p_{AB} = 4001	134
Fig. 6.23	Flow of information in the digital signature procedure	139

LIST OF SYMBOLS

The following is a list of symbols and abbreviations appearing in the thesis and their definitions:

Q	Field of rational numbers
Q_p	Field of p-adic numbers
$\hat{Q}_p$	Segmented p-adic field
Q_g	Ring of g-adic numbers
p, p_{AB}	Prime integers
I_m	Residue system modulo m
(mod m)	Modulo an integer m
$\equiv$	Congruential relation
$\not\equiv$	Incongruence symbol
$a \mid b$	a is a divisor of b
$a \nmid b$	a is not a divisor of b
$\langle a \rangle_b$	Residue of a modulo b
$b^{-1}(m)$ or b^{-1}	Multiplicative inverse of an integer b modulo m
$H(p,r,\alpha)$	Hensel code representation of a rational α
r	Length of segmented p-adic code
$\hat{r}$	Extended p-adic code segment
λ	Period of p-adic expansion
m_α	Mantissa of $H(p,r,\alpha)$
$\tilde{m}_\alpha$	Temporary modified mantissa of $H(p,r,\alpha)$
e_α	Exponent of $H(p,r,\alpha)$
(a,b)	GCD of a,b
Lcm(a,b)	Least common multiple of a,b
Z	Set of all integers
Z^+	Set of positive integers
GF(p)	Galois field with p a prime integer

F_N	Farey sequence of order N
$\lceil x \rceil$	Higher integral part of x(the smallest integer greater than or equal to x)
ϕ	Euler totient function
γ	Maximum value of numerator and denominator in a particular alphabet
α	Primitive element of a prime p

CHAPTER 1

INTRODUCTION

1.1 Historical Background of Number Theory

The theory of numbers is concerned with properties of the natural numbers 1,2,3,4,..., also called the positive integers. These numbers together with the negative integers and zero, form the set of integers Z.

Properties of these numbers have been studied from earliest times. Historical records show that as early as 5700 BC, the ancient Sumerians kept a calendar, and so, they must have developed some form of arithmetic. By 2500 BC, the Sumerians had developed a number system using 60 as a base. The study of numbers evolved through the eras until around 600 BC when the Greeks started the first scientific approach to the study of integers. To them is attributed the true origin of the theory of numbers.

After Euclid, in 300 BC, no significant advances were made in number theory until about AD250 when another Greek mathematician, Diophantus of Alexandria, published 13 books where he made systematic use of algebraic symbols.

Again, after Diophantus, not much progress was made in the theory of numbers until the 17^{th} century when the subject was revived through the efforts of the remarkable French mathematician Pierre de Fermat. He was the first to discover really deep properties of integers.

Then, followed Euler, Lagrange, Legendre, Gauss and Dirichlet who all contributed to the further development of the subject. At the turn of the 18^{th} century, Gauss published his book "Disquisitiones Arithmeticae" which transformed the subject of number theory into a systematic and well founded science.

The field of number theory is vast and some parts require profound knowledge of higher mathematics. Nevertheless, there are many problems in number theory which are very easy to state, yet very difficult to solve [19]. For example, the Goldbach conjecture asserts that every even integer greater than 2 is the sum of two primes. This conjecture was verified up to 100,000 at least, yet no proof for it has ever been provided.

In fact, many such problems deal with prime numbers and this above-mentioned paradox will be reflected in our study of cryptography from chapter 4 onwards.

Number theory has many applications in various fields of science such as in physics, biology, computer science, digital communications, cryptography, etc... [47].

The subject of this thesis is to introduce one area of number theory, which deals with p-adic number systems, in the fast-growing subject of cryptography, thus attempting to design a secure system which achieves the private flow of information between two users in a multi-user, multi-access network.

These two areas of work are introduced in sections 1.2 and 1.3, respectively.

1.2 Introduction to p-adic Number Theory

The set of numbers which are representable in a digital computer in terms of some radix, such as 2(binary), 8(octal) or 10(decimal) is a finite subset of the field of real numbers.

For example, it is not possible to represent a rational number $\frac{a}{b}$ exactly in a radix-β machine if b has a factor relatively prime to β. Thus, $\frac{1}{3}$ cannot be represented exactly in a binary or decimal machine.

Consequently, because of the difficulties associated with using a finite subset to simulate the infinite field of real numbers and trying to solve ill-conditioned problems using inexact arithmetic, it is important to investigate finite number systems which perform exact arithmetic.

This is how attention was turned to p-adic number theory for its possible applications in a digital computer in order to achieve exact computations.

Although Kurt Hensel [21], [22] introduced the p-adic number fields into algebraic number theory in 1908, research into p-adic number systems for error-free computations was only recently initiated by Krishnamurthy [29],[30],[31] and Alparslan [3].

The idea was to truncate the infinite p-adic expansion to a fixed number of digits, r, for all rational numbers in a suitable subset of Q. These fixed-length representations are called Hensel codes.

The finite number system consisting of these Hensel codes has recently been applied to many areas of research such as in matrix processors [29],[30],[31], design of algorithms for error-free computations [18] and in digital signal processing where, very recently, p-adic transformations have been introduced and are currently being investigated [17],[32],[34],[39],[40],[41],[43].

The infinite field of p-adic numbers and the finite-segment p-adic number systems constitute the building blocks of this thesis. They will be considered in detail in chapters 2 and 3 respectively. And, it was found, through their study, that the features of these number systems lend them to a very effective application in the design of two secure cryptographic algorithms. This will be reflected throughout their study in the following chapters when they will be introduced into the subject of cryptography.

1.3 Brief Introduction to Cryptography

The use of secret communications by means of coded messages has been a practice throughout ancient and modern history. In recent wars, codes and ciphers have been used to ensure that secret information was not transmitted to the enemy.

However, it is not only the governments, the military, the security agencies and the diplomatic corps who transmit secret communications. The art of information security is in everyday use. Recent developments in computer science include the concepts of computer-based message systems and electronic mail, amongst others. Modern technology has provided fast and accurate means of transmitting messages. However, there are situations where it is necessary to prevent any intruder or illegal listener from intercepting certain messages on a particular channel. This necessity has transformed the art of secret communications into the science of cryptography, where researchers attempt to design secure systems which prevent cryptanalysts from intercepting and deciphering any coded information being transmitted.

Cryptography relies on various sciences, namely, number theory, complexity theory, computer design and architecture, and the design and analysis of algorithms [20].

These sciences have contributed to the development of the public-key encryption concept where every potential recipient of secret messages publishes his encrypting key, but knowledge of the encrypting key is of no practical help in decryption.

The two schemes proposed in this thesis are based on this concept and will be detailed in chapters 5 and 6. The concept of public-key cryptosystems itself will be studied in great detail in chapter 4.

1.4 Outline of the Thesis

The problem addressed in this thesis was defined in general terms in sections 1.2 and 1.3. In this section, the contents of the thesis are briefly described.

Chapters 2 and 3 consist of a study of p-adic number systems.

In chapter 2, variable-length p-adic number systems are considered. After an introduction to the p-adic fields and the formation of infinite canonic p-adic expansions, Krishnamurthy's algorithm for the conversion from a rational number representation to a representation in the p-adic field Q_p is described. Through the analysis of infinite p-adic sequences, it is found that part of the sequence is recurring. Proof of periodicity and an algorithm for the efficient computation of the p-adic period are provided. Finally, an algorithm for computing variable-length p-adic codes based on the aperiodic and recurrent elements is developed. The algorithm is deterministic in length and thus results in a finite representation of infinite p-adic expansions.

Chapter 3 consists of a thorough study of finite-segment p-adic number systems and the finite p-adic field $\hat{Q}_p$. Bounds on the subset of rational numbers uniquely representable in $\hat{Q}_p$ are given and the reason for their existence is explained. Then, the different structures of Hensel codes are presented for different elements in Q. It was suggested by Krishnamurthy that it was possible to perform

closed p-adic operations in $\hat{Q}_p$. Krishnamurthy's algorithms are presented in detail and their limitations are pointed out. Some of these algorithms lead to erroneous results in certain cases. Modifications to Krishnamurthy's algorithms are suggested and, although it is now possible to perform arithmetic operations in $\hat{Q}_p$, given the original bounds, the closure problem is put into perspective as a practical limitation of the use of finite p-adic number systems. Finally, the different algorithms for converting from $\hat{Q}_p$ back to Q are presented and analysed in detail.

Chapter 4 deals with the mathematical theory of cryptographic systems. An introduction to cryptosystems, the need for them and their features are explained. The notions of authentication and digital signature are also introduced. Then, an introduction to complexity theory and the theory of NP-completeness is given. It is shown that modern public-key cryptosystems are based on problems drawn out of these classes of complexity, and the three main systems which fit into this category are explained in detail.

In chapter 5, the results, modifications and different analyses on p-adic number systems performed in chapters 2 and 3 are linked to the subject of cryptography. After an introduction to the proposed system, p-adic code structures for cryptographic implementation are introduced. This leads to a discussion of the alphabet to be used in the system and the entailing restrictions when generating it. Then, the first realization of the system is performed, based on p-adic fields, and the corresponding encryption and decryption algorithms are detailed. The second realization is based on g-adic

rings and the previous encryption and decryption algorithms are extended to this scheme. However, the decryption algorithm requires the decomposition of g-adic rings into p-adic fields and the algorithm for achieving this transformation is given.

Chapter 6 deals with the detailed evaluation of the proposed system. Cryptanalytic approaches attempting to break it are undertaken. The p-adic scheme is considered first and to which theoretical and statistical attacks are directed. Then, the g-adic scheme is analysed for security and the notions of diffusion and confusion are presented. A new authentication and digital signature procedure is also suggested and, finally, the schemes put forward in this thesis are compared with the three main cryptosystems.

In chapter 7, the contribution made is summarized and suggestions for further research are made.

The listings of computer programs are given in Appendices C1 and C2. The programs are written in FORTRAN 77 and contain, respectively, the variable length p-adic conversion techniques from Q to Q_p, and vice-versa, and a full package implementing p-adic conversion and arithmetic in $\hat{Q}_p$.

CHAPTER 2

VARIABLE-LENGTH p-ADIC NUMBER SYSTEMS

2.1 Infinite p-adic Expansions

In 1908, Hensel [21] introduced the infinite p-adic number system. Out of this system, emerged a major branch of modern algebra called valuation theory. In this section we shall introduce some notions relating to the p-adic valuation, or p-adic norm. However, for a systematic study of valuation theory and p-adic functions, the reader is referred to [4],[6],[27] and [35].

The ordinary absolute function $|\cdot|$ on the field Q has the following basic properties for $\alpha \in Q$:

$$|\alpha| \geq 0 \text{ and } |\alpha| = 0 \text{ if and only if } \alpha = 0 \qquad (2.1)$$

For α and $\beta \in Q$,

$$|\alpha\beta| = |\alpha||\beta| \qquad (2.2)$$

$$|\alpha+\beta| \leq |\alpha| + |\beta| \qquad (2.3)$$

It is noticed that property (2.3) is the "triangle inequality".

In the field of rational numbers Q, the absolute value mapping, $|\cdot|$, can be shown to be norm on Q, [4]. Another norm on Q, of more interest to us here, can be constructed on the observation that, if α is a non-zero element of Q, then α can be expressed uniquely in the form

$$\alpha = \frac{a}{b} \cdot p^n \qquad (2.4)$$

where p is a prime

$a,b,\ n \in Z, \qquad b \neq 0$

$(a,b) = 1 \qquad$ and $\qquad p \nmid a,\ p \nmid b.$

With this definition, the p-adic norm can then be defined [27] such that:

$$|\alpha|_p = \begin{cases} p^{-n} & \text{if} \quad \alpha \neq 0 \\ 0 & \text{if} \quad \alpha = 0 \end{cases} \qquad (2.5)$$

and it should be observed that the p-adic norm is counter-intuitive since a large positive integer n implies a small value for the p-adic norm.

The concept of the p-adic field Q_p follows from the principle that any rational number α expressed as in (2.4) can be uniquely represented by an infinite series of the form:

$$\alpha = \sum_{n=-m}^{\infty} a_n p^n \qquad ; \qquad a_n \in I_p, \quad m \in Z \qquad (2.6)$$

and this infinite series converges to α with respect to the p-adic norm [6].

It is convenient to introduce a shorthand notation for (2.6) similar to the decimal representation used for base 10. In a conventional decimal expansion, there are infinitely many digits corresponding to the negative powers of the radix 10, whereas a p-adic sequence is composed of infinitely many digits representing the non-negative powers of p. For convenience in arithmetic operations, these digits will be written to the right of the p-adic point. Hence, the p-adic series in (2.6) will be expressed in the form:

$$\alpha = a_{-m}\ a_{-m+1} \ \ldots\ a_{-1} \cdot a_0\ a_1 \ \ldots\ (p) \qquad (2.7)$$

It is also noted that the digit positions corresponding to the non-negative and negative powers of the base are reversed for p-adic expansions and p-ary expansions. For example, if α is a positive integer, then its p-adic representation is of the form:

$$\alpha = \cdot a_0\ a_1 \ \ldots\ a_k\ 00 \ \ldots\ \ (p) \qquad (2.8)$$

while its p-ary representation is the reflection of (2.8) about the p-adic point:

$$\alpha = [a_k\ a_{k-1} \ \ldots\ a_0 \cdot]_p \qquad (2.9)$$

Also, it is observed [29] that negative rationals occur as true-complement (left-to-right) of the positive number. This and other word formats will be discussed in more detail in chapter 3 when we consider segmented p-adic codes and finite-length p-adic arithmetic.

In order to convert a rational number α to p-adic form, we will present an algorithm developed by Krishnamurthy [29] and [31]. In section 2.3, an iterative algorithm will be developed which allows to detect the period in the p-adic expansion and to determine the "aperiodic" elements in the expansion.

2.2 Krishnamurthy's Algorithm for Conversion from Rational Number to Infinite p-adic Form

Given a rational $\alpha = \frac{a}{b}$ with $(a,b) = 1$ and $b \neq 0$, then the p-adic expansion can be obtained by the following algorithm:

1. Set $c = a$

   Check if $p^n | b$. The value of n will control the position of the p-adic point.

   Set $d = b/p^n$.

2. Solve the congruence:

   $$dx \equiv 1 \pmod{p}$$

   If x_n is a solution, then

   $$a_n \equiv c \cdot x_n \pmod{p}.$$

3. Set $\gamma = \frac{c}{d} - a_n$

   $$= \frac{c'}{d'} .$$

4. If $\gamma = 0$, set $a_i = 0$ for $i > n$ and stop.

The value of c' will be divisible by p. Set $c = \frac{c'}{p}$.

Set d = d'

Go to step 2.

Note that, contrary to the algorithm which will be presented in the next section, this algorithm is non-terminating since the convergence to the given rational can be achieved only for an infinite number of digits in the expansion, if the rational is neither an integer nor a radix fraction.

2.3 Period Detection

In this section we will show that the canonical p-adic expansion of a rational number is periodic (or recurring). An algorithmic approach to computing the period will be given, followed by an algorithm for the determination of the aperiodic elements and the elements comprised in one recurring cycle.

2.3.1 Proof of Periodicity in Infinite p-adic Number Systems

If $\alpha = \frac{a}{b}$ where (a,b) = 1, then α can be written in the form

$$\alpha = \frac{a}{d \cdot p^k} \qquad (2.10)$$

where $p \nmid d$ and $k \in Z$.

Since p is relatively prime to d taken as modulus, then in the series of powers of p:

$$P = \{1, p, p^2, p^3, \ldots\} \tag{2.11}$$

all terms are relatively prime to d and, hence, are congruent to terms of a reduced system of residues modulo d [1].

And since any reduced system of residues modulo d contains (d-1) terms whereas the series {P} is infinite, then there must exist two terms in {P}, say p^i and p^j congruent modulo d:

$$p^i \equiv p^j \pmod{d} \tag{2.12}$$

Assuming i > j, then by dividing both sides of the congruence by p^j, which is relatively prime to d, we get

$$p^{i-j} \equiv 1 \pmod{d} \tag{2.13}$$

and so there are positive exponents, k, for which

$$p^k \equiv 1 \pmod{d} \tag{2.14}$$

Let λ be the smallest of such exponents. So, by its very definition, λ is characterized by two properties:

1. $p^{\lambda} \equiv 1 \pmod{d}$ (2.15)

2. No power of p with positive exponent $< \lambda$ is congruent to 1 (mod d).

Now, if k is any positive exponent for which congruence (2.14) is satisfied, then

$$k \equiv 0 \pmod{\lambda} \quad . \tag{2.16}$$

This is so since, if we assume that, conversely, k is not divisible by λ, then we can write

$$k = \lambda q + r \tag{2.17}$$

where $q \in Z^{+}$

and $0 < r < \lambda$.

Then

$$\begin{aligned} p^{k} &= p^{(\lambda q + r)} \\ &= (p^{\lambda})^{q} \cdot p^{r} \end{aligned} \tag{2.18}$$

But, from (2.15),

$$p^{\lambda} \equiv 1 \pmod{d}.$$

Hence

$$p^k = p^r \equiv 1 \pmod{d} \qquad (2.19)$$

and this cannot hold unless $r = 0$, so that k is divisible by λ and (2.16) is true.

Hence, we can say that the elements

$$P = \{1, p, p^2, p^3, \ldots, p^{\lambda-1}\} \qquad (2.20)$$

are all different modulo d, and that the terms in the sequences:

$$P' = \{p^{\lambda}, p^{\lambda+1}, \ldots, p^{2\lambda-1}\}$$

$$P'' = \{p^{2\lambda}, p^{2\lambda+1}, \ldots, p^{3\lambda-1}\}$$

$$\vdots$$

and so on

are congruent modulo d to the terms in $\{P\}$, i.e. the series

$$P = \{1, p, p^2, \ldots, p^{\lambda-1}\}$$

is periodic with a period length λ.

2.3.2 Algorithm for the Period Computation

Having established from section 2.3.1 that the infinite p-adic expansion of a rational number has a periodicity λ, the problem now is to develop a practical algorithm for the solution of the congruence

$$p^{\lambda} \equiv 1 \pmod{d}$$

to determine the effective value of λ.

The following is a more detailed analysis of the properties of the congruence

$$a^x \equiv 1 \pmod{m} \tag{2.21}$$

These properties will lead to the derivation of the required algorithm.

The least positive integral solution $x = k$ of the congruence (2.21) is called the exponent to which a belongs modulo m [5].

Let $\phi(m)$ be the Euler totient function of m. The following properties of ϕ makes it possible to calculate $\phi(m)$ [9]:

1. $$\phi(mn) = \phi(m)\,\phi(n) \quad \text{whenever } (m,n) = 1 \tag{2.22}$$

2. $$\phi(q) = q - 1 \quad \text{for } q \text{ prime} \tag{2.23}$$

3. $$\phi(q^k) = q^k - q^{k-1}$$
$$= q^{k-1}(q-1) \quad \text{for } q \text{ prime} \tag{2.24}$$

4. $$\phi(m) = m \prod_{q \mid m} \left(1 - \frac{1}{q}\right) \tag{2.25}$$

We also state, without proof, Euler's theorem and Fermat's corollary (Fermat's little theorem). Their proofs can be found in [5]:

Theorem: Let a,m be integers with $m > 0$. If $(a,m) = 1$, then

$$a^{\phi(m)} \equiv 1 \pmod{m} \tag{2.26}$$

Corollary: If q is a prime and a is an integer such that $q \nmid a$, then

$$a^{q-1} \equiv 1 \pmod{q} \tag{2.27}$$

Also, an integer a is called a primitive root modulo m if x in (2.21) is equal to $\phi(m)$ (i.e. congruence, (2.26)).

Going back to the original congruence (2.15), we can write d such that

$$d = p_1^{\varepsilon_1} \; p_2^{\varepsilon_2} \; \ldots \; p_n^{\varepsilon_n}$$

$$\therefore \quad d = \prod_{i=1}^{n} p_i^{e_i} \tag{2.28}$$

where $p_i \neq p$, $\forall i$. This is the prime factor decomposition of d. Hence,

$$p^{\lambda} \equiv 1 \pmod{p_1^{\varepsilon_1} \cdot p_2^{\varepsilon_2} \cdot \ldots p_n^{\varepsilon_n}} \tag{2.29}$$

The Chinese Remainder Theorem (CRT), [42], states that solving (2.29) is equivalent to solving the simultaneous congruences

$$p^{\lambda_i} \equiv 1 \pmod{p_i^{e_i}} \; ; \qquad i = 1,2,\ldots,n \tag{2.30}$$

and hence,

$$\lambda = \mathrm{Lcm}(\lambda_i) \tag{2.31}$$

Recalling from section 2.3.1, congruence (2.16) shows that, if $(p,d) = 1$ then

$$p^k \equiv 1 \pmod{}$$

if and only if $\lambda | k$.

Consequently, we can write that

$$\lambda | \phi(d) \tag{2.32}$$

and that

$$\lambda_i | \phi(p_i^{e_i}) \tag{2.33}$$

However, using (2.24), $\phi(p_i^{e_i})$ can be factored into a product of powers of distinct primes, say

$$\begin{aligned} \phi(p_i^{e_i}) &= p_i^{e_i - 1} (p_i - 1) \\ &= q_1^{\eta_1} \cdot q_2^{\eta_2} \cdots q_k^{\eta_k} \end{aligned} \tag{2.34}$$

and the algorithm is based on the above factorization procedure. The idea is to divide $\phi(p_i^{e_i})$ by all prime factors q_j such that $\phi(p_i^{e_i})/q_j$ is a solution of (2.15) to obtain the least integral solution.

Function 'PERIOD' in Appendix C1 reflects the theoretical analysis performed in this section.

Table 2.1(A) gives the different values of λ for various combinations of the primes p and denominators d and compares these values with the Euler ϕ-function. In Table 2.1(B), larger values of p are considered for the various denominators d.

Table 2.1(A): Comparison of period lengths with the Euler ϕ-function for various primes p and denominators d.

d	$\phi(d)$	p						
		5	7	11	13	29	37	53
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	2	2	1	2	1	2	1	2
4	2	1	2	2	1	1	1	1
5	4	1	4	1	4	2	4	4
6	2	2	1	2	1	2	1	2
7	6	6	1	3	2	1	3	3
8	4	2	2	2	2	2	2	2
9	6	6	3	6	3	6	1	2
10	4	1	4	1	4	2	4	4
11	10	5	10	1	10	10	5	5
12	4	2	2	2	1	2	1	2
13	12	4	12	12	1	3	12	1
14	6	6	1	3	2	1	3	3
15	8	2	4	2	4	2	4	4
16	8	4	2	4	4	4	4	4
17	16	16	16	16	4	16	16	8
18	6	6	3	6	3	6	1	2
19	18	9	3	3	18	18	6	18
20	8	1	4	2	4	2	4	4

Table 2.1(B): Comparison of period lengths with the Euler ϕ-function for various larger primes p and the same denominators d

d	$\phi(d)$	p						
		353	1297	2909	4663	6481	8389	9973
1	1	1	1	1	1	1	1	1
2	1	1	1	1	1	1	1	1
3	2	2	2	2	2	2	2	2
4	2	2	2	1	2	2	2	2
5	4	4	4	4	4	4	4	4
6	2	2	2	2	2	2	2	2
7	6	6	6	6	6	6	6	6
8	4	4	4	2	4	1	4	4
9	6	6	6	2	6	6	6	6
10	4	4	4	4	4	4	4	4
11	10	10	10	10	10	10	10	10
12	4	2	2	2	2	2	2	2
13	12	12	12	12	12	12	12	12
14	6	6	6	6	6	6	6	6
15	8	4	4	4	4	4	4	4
16	8	8	8	4	4	8	8	8
17	16	16	16	16	8	16	16	16
18	6	6	6	2	6	6	6	6
19	18	18	18	18	18	18	18	18
20	8	4	4	4	4	4	4	4

2.4 Algorithm for Computing the Variable-Length p-adic Expansion

In section 2.2 we presented a non-terminating algorithm, developed by Krishnamurthy, for the computation of the infinite p-adic expansion of a rational number α.

In this section, another algorithm based on a lemma by Mahler [35] is developed and which puts into perspective the notion of p-adic period presented in the previous sections. We thus obtain a variable-length p-adic expansion where the aperiodic terms are computed and the recurring elements are determined only once.

By the unique factorization theorem in the domain of integers, any rational number α can be written uniquely in the form:

$$\alpha = p^k \cdot \frac{a}{b} \tag{2.35}$$

where $k \in Z$, $(a,b) = 1$, $p \nmid a$, $p \nmid b$.

On the other hand, the infinite p-adic expansion of a p-integral rational number

$$\frac{a}{b} = \sum_{n=0}^{\infty} a_n \, p^n \qquad , \qquad a_n \in I_p \tag{2.36}$$

can be computed iteratively using the formulas

$$\frac{a}{b} = a_0 + a_1 p + \ldots + a_{n-1} p^{n-1} + \frac{r_n}{b} \cdot p^n \qquad (2.37)$$

where $r_0 = a$

and $\frac{r_n}{b} = a_n + \frac{r_{n+1}}{b} \cdot p$

Hence,

$$r_n = a_n b + r_{n+1} p$$

or

$$r_{n+1} = \frac{r_n - a_n b}{p} \qquad (2.38)$$

But, since our analysis is concerned with a residue system modulo p, then equation (2.38) can be written in the form:

$$\langle a_n\ b \rangle_p = \langle r_n - r_{n+1}\ p \rangle_p$$

Hence,

$$\langle a_n\ b \rangle_p = \langle r_n \rangle_p \qquad (2.39)$$

Also, since $b \nmid P$, the multiplicative inverse of b modulo p exists, and since $a_n \in I_p$, we have:

$$a_n = \langle a_n \rangle_p = \langle c \cdot r_n \rangle_p \qquad (2.40)$$

where

$$c = b^{-1}(p) \tag{2.41}$$

or

$$bc \equiv 1 \pmod{p}$$

Setting $r_0 = a$ and using equations (2.38) - (2.41), the different a_n are computed and then the subsequent r_{n+1} are computed from a knowledge of a_n and r_n.

If

$$r_{n+1} = r_i$$

for any $i = 1,2,\ldots,n$, then the terms

$$\{a_0, a_1, \ldots, a_{i-1}\}$$

are the aperiodic elements, whereas the terms

$$\{a_i, a_{i+1}, \ldots, a_{n-1}, a_n\}$$

represent the periodic elements with periodicity

$$\lambda = n - i + 1 \tag{2.42}$$

It is now clear that it is possible to obtain a deterministic variable-length representation of an otherwise infinite p-adic expansion. There remains the problem of converting such a code back to its rational form. We will postpone our analysis of this problem until we discuss finite-segment p-adic number systems and, especially, the arithmetic in $\hat{Q}_p$.

CHAPTER 3

FINITE-SEGMENT p-ADIC NUMBER SYSTEMS

3.1 Mathematical Introduction to Finite p-adic Number Systems

In chapter 2 an analysis of the p-adic field Q_p was performed. In this chapter we consider the segmented p-adic field $\hat{Q}_p$ where a unique code, called Hensel code, is derived for a rational number by truncating its infinite p-adic expansion to a finite number of digits r.

The uniqueness of this code, however, spans over a certain range of rationals. This range is determined by the order-N Farey fractions [18], F_N, where the integer N is the largest positive integer which satisfies the inequality

$$m \geq 2N^2 + 1 \tag{3.1}$$

where

$$m = p^r \tag{3.2}$$

for a prime p.

Hence

$$N = \checkmark \frac{p^r - 1}{2} \tag{3.3}$$

And so, for a rational $\alpha = \frac{a}{b}$ to have a unique Hensel code $H(p,r,\alpha)$, the following bounds must be satisfied:

$$-\sqrt{\frac{p^r-1}{2}} \leq a,b \leq \sqrt{\frac{p^r-1}{2}} \tag{3.4}$$

where $b \neq 0$ and $p \nmid b$.

These bounds were never mathematically justified [32]. However, using Hensel codes to represent rationals is essentially the same as calculating modulo p^r. Consider the two rational numbers $\alpha = \frac{a}{b}$ and and $\beta = \frac{c}{d}$. If $H(p,r,\alpha) = H(p,r,\beta)$, then

$$p^r | (ad - bc) \tag{3.5}$$

But, if a,b,c,d are bounded by (3.4), then

$$|ad| , |bc| \leq \frac{p^r-1}{2} \tag{3.6}$$

Therefore

$$|ad - bc| \leq p^r-1 < p^r \tag{3.7}$$

But (3.5) and (3.7) are contradictory statements and, in fact, the only integer whose absolute value is less than p^r which is divisible by p^r is zero. Hence,

$$ad = bc \quad \text{or} \quad \frac{a}{b} = \frac{c}{d} \tag{3.8}$$

and the rationals α and β are the same.

3.2 Structure of Hensel Codes

In [29] and [31], Krishnamurthy introduced the finite-segment Hensel codes in $\hat{Q}_p$. Having satisfied the bounds in (3.4), the Hensel code is then unique. Furthermore, the individual digit positions have the following positive or negative fractional weights :

$$p^0,\ p^1,\ p^2\ ,\ldots,\ p^{r/2-1},\ \frac{-p^{r/2}}{p^{r/2-1}}\ ,\ \frac{-p^{r/2+1}}{p^{r/2-1}}\ ,\ldots,\ \frac{-p^{r-1}}{p^{r/2-1}} \qquad (3.9)$$

These fractional weights, according to [29], are necessary to convert Hensel codes back to their rational form. However, as will be described later in section 3.4, this property is not of paramount importance if other conversion techniques are to be used.

On the other hand, the sequence of weights (3.9) shows that the length, r, of any Hensel code must be even. This, again, will be put into perspective in section 3.4. It enables us, however, to formulate the structure of Hensel codes for $\alpha \in F_N$.

In chapter 2, it was shown that the infinite p-adic expansion of α can be written as:

$$\alpha = a_{-n}, a_{-n+1},\ldots,a_{-1}\cdot a_0,\ldots,a_m,a_{m+1},\ldots\ (p) \qquad (3.10)$$

Truncating the above expansion gives

$$H(p,r,\alpha) = a_{-n}, a_{-n+1},\ldots,a_{-1}\cdot a_0,a_1,\ldots,a_m\ (p) \quad (3.11)$$

which is of length

$$r = n + m + 1 \quad (3.12)$$

The elements a_i, $i = -n,\ldots,m$ can be determined according to the algorithm given in section 2.2, chapter 2. In this case, however, the algorithm is made to terminate when all r terms of the truncated expansion have been computed.

As a consequence of (3.4) and the fact that the total number of digits, r, in the code must be even, positive and negative integers are represented by exactly $\frac{r}{2}$ digits, assuming that the p-adic point is to the left of the zeroth digit, the remaining digits being 0 or (p-1) respectively.

By shifting the p-adic point to the right of the $0,1,2,\ldots,\frac{r}{2} - 1^{th}$ position, the above integers become radix fractions, i.e., fractions the denominator of which is divisible by p^n and, consequently, all radix fractions whose denominators do not exceed N become representable. All other rationals will, in general, occupy all the r positions.

Hence, positive integers will be in the form:

$$\cdot a_0, a_1,\ldots,a_{r/2-1}, \underbrace{0,0,\ldots,0}_{r/2-1} \quad (3.13)$$

e.g., H(5,6,87) = .223000.

Negative integers, on the other hand, are in complement form, and have the following structure:

$$\cdot a_0, a_1, \ldots, a_{r/2-1}, \underbrace{(p-1),(p-1),\ldots,(p-1)}_{r/2-1} \qquad (3.14)$$

e.g., H(5,6,-73) = .202444.

Radix fractions are obtained by moving the p-adic point of positive or negative integers to the right by a number of locations $\leq \frac{r}{2} - 1$;

e.g., H(5,6,87/5) = 2.23000

H(5,6,87/25) = 22.3000

H(5,6,-73/25) = 20.2444.

Soft and hard fractions have the general form of the segmented p-adic number representation given in (3.11) and the digits assume values between 0 and p-1.

3.3 Arithmetic Operations in Segmented p-adic Fields

Throughout this section, and merely for convenience purposes, the Hensel code H(p,r,α) will be denoted as an ordered pair in mantissa-exponent form, (m_α, e_α) [31]. And, since the length of the

code is fixed to r digits, e_α is allowed to have the value 0 or negative values.

When $e_\alpha = -n$, the p-adic point is then placed n digits to the right of the left-most digit of m_α. Consequently, the mantissa is of the form

$$m_\alpha = 0 \cdot a_0\ a_1\ \ldots\ a_{r-1} \tag{3.15}$$

and

$$e_\alpha \leq 0 \tag{3.16}$$

When performing arithmetic operations in $\hat{Q}_p$, it is necessary to ensure that the closure property is maintained throughout our computation. That is, if the operands are within the bounds (3.4), then so must be the result of the operation.

In the following sections (3.3.1 - 3.3.4), we present the algorithms developed by Krishnamurthy [29] and [31] for the four basic operations. In section 3.3.5 we show fundamental limitations in Krishnamurthy's algorithms and give our corresponding modified algorithms.

3.3.1 p-adic Addition in $\hat{Q}_p$

Given $\alpha = \frac{a}{b}$ and $\beta = \frac{c}{d}$, such that $b,d \neq 0$, $(a,b) = 1$ and $(c,d) = 1$, then the corresponding Hensel codes of α and β of the same length r in the segmented p-adic field $\hat{Q}_p$ may be written

respectively as:

$$H(p,r,\alpha) = (m_\alpha, e_\alpha)$$

$$H(p,r,\beta) = (m_\beta, e_\beta)$$

where

$$\left.\begin{aligned} m_\alpha &= 0{\cdot}a_0\ a_1 \ \ldots\ a_{r-1} \\ m_\beta &= 0{\cdot}b_0\ b_1 \ldots\ b_{r-1} \end{aligned}\right\} \quad (3.17)$$

with r even, and e_α and e_β both satisfying the inequality:

$$-\left(\frac{r}{2} - 1\right) \leq e_\alpha, e_\beta < 1 \quad (3.18)$$

The algorithm for adding $H(p,r,\alpha)$ and $H(p,r,\beta)$ retains the lower exponent and finds the sum digit s_i and carry digit c_{i+1} from a knowledge of a_i, b_i and c_i. Thus

$$s_i \equiv (a_i + b_i + c_i) \pmod{p} \quad (3.19)$$

$$\text{for } i = 1,2,\ldots,r-1$$

with $c_o = 0$

and $c_{i+1} = 1$ if $(a_i + b_i + c_i) \geq p$

$= 0$ otherwise

and ignoring c_r.

e.g., $H(5,4,3/13) + H(5,4,11/13) = (0.1143,0) + (0.2430,0)$

$= (0.3034,0)$

$= H(5,4,14/13)$.

3.3.2 p-adic Substraction in $\hat{Q}_p$

p-adic substraction is realized as a complemented p-adic addition. Thus, to perform the operation $H(p,r,\alpha) - H(p,r,\beta)$, we write,

$$\begin{aligned} H(p,r,\alpha) - H(p,r,\beta) &= H(p,r,\alpha) + H(p,r,-\beta) \\ &= H(p,r,\alpha) + H(p,r,\bar{\beta}) \\ &= (m_\alpha, e_\alpha) + (\bar{m}_\beta, e_\beta) \end{aligned} \qquad (3.20)$$

where

$$\bar{m}_\beta = 0.\bar{b}_0\ \bar{b}_1 \ \ldots\ \bar{b}_{r-1}$$

is the complement of

$$m_\beta = 0.b_0\ b_1 \ \ldots\ b_{r-1}$$

and the elements $\bar{b}_i$ are obtained through the following rules :

1) If $b_i \neq 0$ for $0 \leq i \leq (r-1)$

then $\bar{b}_i = p - b_i$ for $i = 0$

and $\bar{b}_i = (p-1) - b_i$ for $1 \leq i \leq (r-1)$

2) If $b_i = 0$ for $0 \leq i \leq j$

then $\bar{b}_i = 0$ for $0 \leq i \leq j$

and $\bar{b}_{j+1} = p - b_{j+1}$

$\bar{b}_i = (p-1) - b_i$ for $(j+2) \leq i \leq (r-1)$

3.3.3 p-adic Multiplication in $\hat{Q}_p$

The multiplication algorithm of $H(p,r,\alpha)$ and $H(p,r,\beta)$ consists in forming the cross-products of the mantissa such that:

$$P_{i,j} = b_i \, a_j \quad , \quad \text{for} \quad 0 \leq i \leq (r-1) \quad \text{and} \quad j = 0,1,\ldots,r-1 \qquad (3.21)$$

and then forming the partial products P_i and the final product P by successive shifts and additions as given by:

$$P_i = \sum_{j=0}^{r-1} P_{i,j} \, \Delta(j) \qquad (3.22)$$

and

$$P = \sum_{i=0}^{r-1} P_i \, \Delta(i) \qquad (3.23)$$

where $\Delta(x)$ denotes a right shift of the partial result by x digits . The exponent of the result is then $(e_\alpha + e_\beta)$.

3.3.4 p-adic Division in $\hat{Q}_p$

Consider the finite-segment p-adic division of $H(p,r,\alpha)$, the dividend, by $H(p,r,\beta)$, the divisor. The quotient is then $H(p,r,\gamma)$ such that:

$$m_\alpha = 0 \cdot a_0\ a_1\ \cdots\ a_{r-1}$$

$$m_\beta = 0 \cdot b_0\ b_1\ \cdots\ b_{r-1}$$

$$m_\gamma = 0 \cdot q_0\ q_1\ \cdots\ q_{r-1}.$$

The division operation, therefore, is similar to p-adic multiplication but for computing the multiplicative inverse of b_o, $b_o^{-1}(p)$. Algorithms for computing $b_o^{-1}(p)$ can be found in [25].

If $b_o = 0$, the divisor is shifted left, keeping count, until the first non-zero digit and the exponent e_β is adjusted accordingly.

The algorithm for division is then as follows:

Set $R_o = m_\alpha$. This represents the zeroth partial remainder. Let R_i denote the partial remainder at the i^{th} stage and R_{ii} denote its i^{th} digit. Then,

$$q_i \equiv R_{ii}\ b_o^{-1} \pmod p \quad \text{for } i = 0,1,2,\ldots,r-1 \qquad (3.24)$$

The next partial remainder R_{i+1} is then given by

$$R_{i+1} = R_i + q_i \cdot \bar{m}_\beta \cdot \Delta(i) \qquad (3.25)$$

where $\bar{m}_\beta$ is the complement of m_β and $\Delta(i)$ denotes a right shift by i digits, and the algorithm terminates when q_{r-1} is obtained. The exponent e_γ of the result is then $(e_\alpha - e_\beta)$.

3.3.5 Limitations of Krishnamurthy's Algorithms

In this section we show that the algorithms for addition and multiplication (and hence, those for substraction and division) presented earlier do not always generate a correct result having the same code-word-length as the two operands. In fact, in such cases, confining these operations to the prescribed size without incurring any modifications to the algorithms, results in an incomplete code which is seriously erroneous [16]. After discussing these limitations we will present the modified complete algorithms for closed finite p-adic arithmetic operations in $\hat{Q}_p$.

3.3.5.1 Limitations of the Addition Algorithm

The algorithm described in section 3.3.1 would always work if the exponents were different, that is,

$$e_\alpha \neq e_\beta \tag{3.26}$$

Alternatively, if the exponents were equal, then the algorithm is guaranteed to work if they were both zero. On the other hand, if e_α and e_β were equal but are not zero, i.e., $p^n | b,d$, then the algorithm would work if and only if $s_i \not\equiv 0 \pmod{p}$, where i denotes the position of any leading zero in the sum.

Summarizing, the algorithm for addition would yield the correct result in all instances except in those cases where:

$$\left.\begin{aligned} & e_\alpha = e_\beta < 0 \\ \text{and} \quad & s_i \equiv 0 \pmod{p}, \quad i = 0, 1, \ldots, k \\ & \text{where } k \leq |e_\alpha| - 1 \end{aligned}\right\} \tag{3.27}$$

In this latter case, the resulting sum, which should correspond to m_γ (where $\gamma = \alpha+\beta$), contains k+1 leading zeros. These leading zeros not only are meaningless, exactly as in p-ary arithmetic, but their existence leads to an entirely erroneous code. Also, in what regards the final exponent, e_γ, it is not equal to e_α or e_β as shown in the algorithm.

If these leading zeros were simply discarded from the resulting mantissa, then, consequently, $H(p,r,\gamma)$ is no longer a correct Hensel code of length r, but its effective length is r-k-1 and the missing digits and the correct exponent, e_γ, are yet to be determined for a complete exact code.

3.3.5.2 Modified Addition Algorithm

To overcome the problem mentioned in section 3.3.5.1, in the case where $e_\alpha = e_\beta < 0$, the algorithm presented in section 3.3.1 should be carried over the same range, r, until s_r is reached where it should be extended as follows:

1. Compute c_r

2. If $s_i \equiv 0 \pmod p$, $i = 0,1,\ldots,k$ where $k \leq |e_\alpha| - 1$,
   then compute $H(p,\hat{r},\alpha)$ and $H(p,\hat{r},\beta)$
   where $\hat{r} = r+2$ for $k = 0$
   $= r + k$ k even
   $= r+k+1$ k odd

3. $s_i \equiv (a_i + b_i + c_i) \pmod{p}$, $i = r, r+1, \ldots, r+k$, taking the value of c_r, computed in (1), into consideration.

4. The resulting sequence of digits is shifted left by k+1 locations yielding m_γ, and $e_\gamma = e_\alpha + k + 1$. This code is the required Hensel code of size r.

Example: Consider the segmented p-adic addition of $\frac{11}{10}$ and $\frac{16}{15}$. Their corresponding Hensel codes for p = 5 and r = 4 are:

$H(5,4,11/10) = (0.3322,-1)$

$H(5,4,16/15) = (0.2413,-1)$.

If we apply Krishnamurthy's algorithm of section 3.3.1, as it stands, the segmented p-adic operation would yield the result (0.0340,-1), which does not correspond to the Hensel code of $\gamma = 13/6$. If fact, it is not a Hensel code at all.

Consider now the modified addition algorithm discussed above.

Since $e_\alpha = e_\beta = -1$ and $s_o = 0$, where k = 0, then $\hat{r} = 6$. Thus, H(5,6,11/10) + H(5,6,16/15) results in a mantissa of 0.03404 where the addition process is stopped after reaching $i = r+k = 4$ (whereas it was terminated at $i = r-1 = 3$ in Krishnamurthy's algorithm).

Now, shifting this mantissa left by 1 location yields $m_\gamma = 0.3404$ and $e_\gamma = 0$, i.e.,

$$H(5,4,\gamma) = (0.3404,0)$$

which corresponds to $\gamma = 13/6$.

3.3.5.3 Limitations of the Multiplication Algorithm

Here, again, we point out that Krishnamurthy's algorithm for the p-adic multiplication in $\hat{Q}_p$, described in section 3.3.3 would perform correctly provided either one of the following two conditions is satisfied:

1. $e_\alpha = e_\beta$

2. $e_\alpha \neq e_\beta$ such that $e_\alpha < e_\beta$ and m_β does not consist of any leading zeros in its first r/2 digits.

If we consider now the case where

$$e_\alpha < e_\beta$$

and

$$m_{\beta_i} = 0, \quad i = 0,1,\ldots,k \quad \text{where } k \leq \left(\frac{r}{2} - 1\right),$$

then, unavoidably, and after performing the described algorithm, leading zeros will appear in the final product leaving its mantissa, m_γ, incomplete, with the exponent, e_γ, of the product not equal to $e_\alpha + e_\beta$ as stipulated in section 3.3.3.

3.3.5.4 Modified Multiplication Algorithm

The erroneous code resulting from applying Krishnamurthy's algorithm as it stands can be corrected if the following preliminary steps were undertaken

1. If $m_{\beta_i} = 0, \quad i = 0,1,\ldots,k, \quad$ where $k \leq \left(\frac{r}{2} - 1\right)$,

   then compute $H(p,\hat{r},\beta)$ where, again,

$$\hat{r} = r + 2 \qquad \text{for } k = 0$$
$$= r + k \qquad k \text{ even}$$
$$= r + k + 1 \qquad k \text{ odd}$$

2. Shift m_β of $H(p,\hat{r},\beta)$ left by k+1 locations resulting in the temporary shifted sequence $\tilde{m}_\beta$. It should be noted that the effective length of $\tilde{m}_\beta$ under consideration is taken equal to r, but does not correspond to the mantissa of any Hensel code.

From this point onwards, Krishnamurthy's algorithm is applied with the difference that the product's exponent, e_γ, is given by:

$$e_\gamma = e_\alpha + e_\beta + k + 1$$

and this exponent, together with the resulting mantissa, correspond to the Hensel code of $\gamma = \alpha \cdot \beta$.

We illustrate the points discussed in section 3.3.5.3 and in this section by the following example.

Example: Assume $\alpha = 1/15$ and $\beta = 5/4$. Then, the multiplication of $H(p,r,\alpha)$ by $H(p,r,\gamma)$ consists in multiplying the Hensel codes $(0.2313,-1)$ and $(0.0433,0)$ for $p = 5$ and $r = 4$.

The segmented p-adic multiplication algorithm as described in section 3.3.3 is depicted in Table 3.1 and the product resulting is, thus, $H(5,4,\gamma) = (0.0342,-1)$ which, again, is obviously erroneous.

However, since $e_\alpha < e_\beta$ and the first leading digit in $m_\beta = 0$, at $k = 0$, then $H(5,6,5/4) = (0.043333,0)$ and $\bar{m}_\beta = 0.4333$.

Given this temporary value of the multiplier mantissa, the segmented p-adic multiplication is performed according to Krishnamurthy's algorithm and as depicted in Table 3.2.

Hence, $m_\gamma = 0.3424$

and $e_\gamma = -1 + 0 + 0 + 1 = 0$

and the final product $H(5,4,\gamma) = (0.3424,0)$ which corresponds to $\gamma = 1/12 = 1/15 \cdot 5/4$.

Table 3.1: Krishnamurthy's algorithm for segmented p-adic multiplication in the case of $\alpha = \frac{1}{15}$ and $\beta = \frac{5}{4}$ for $p = 5$ and $r = 4$

i,j	...	0	1	2	3
a_j	...	2	3	1	3
b_i	...	0	4	3	3
$P_{0,0}$	...	0	0		
$P_{0,1}$	...		0	0	
$P_{0,2}$	...			0	0
$P_{0,3}$	...				0
P_0	...	0	0	0	0
.					
P_1	...		3	3	1
.					
P_2	...			1	0
.					
P_3	...				1
P	...	0	3	4	2

Table 3.2: Modified algorithm for segmented p-adic multiplication in the case of $\alpha = \frac{1}{15}$ and $\beta = \frac{5}{4}$ for $p = 5$ and $r = 4$

i,j	...	0	1	2	3
a_j	...	2	3	1	3
b_i	...	4	3	3	3
$P_{0,0}$	...	3	1		
$P_{0,1}$	...		2	2	
$P_{0,2}$	...			4	0
$P_{0,3}$	...				2
P_0	...	3	3	1	3
.					
P_1	...		1	0	0
.					
P_2	...			1	0
.					
P_3	...				1
P	...	3	4	2	4

3.4 Conversion of Hensel Codes and Variable-Length p-adic Expansions to Rational Form

Having analysed the arithmetic in $\hat{Q}_p$, we now consider the problem of converting a finite or infinite p-adic code to its rational equivalent. There are three main conversion techniques [29] which we will analyse in turn.

3.4.1 Method of Congruences Based on p-adic Weights

In section 3.2 it was mentioned that the digit positions in a p-adic representation have corresponding fractional weights given by (3.9).

The conversion from p-adic form to rational form depends on the type of fraction under consideration:

a) Soft fractions: These are rational fractions a/b where $b|(p^{r/2}-1)$. The weighted sum W will then be equal to the actual value of such particular rational and satisfies the equation:

$$\frac{a}{b} = \frac{W}{p^{r/2} - 1}$$

or $$a(p^{r/2} - 1) - bW = 0 \qquad (3.28)$$

and hence the conversion from a p-adic representation to rational form only involves finding this weighted sum.

b. Hard fractions: Fractions which do not have the property that the denominator is a divisor of $p^{r/2}-1$ are called hard fractions. Their weighted sum will not be equal to their actual value and, consequently, such weights are called pseudo-weights.

Such fractions will only satisfy the following congruence:

$$a(p^{r/2}-1) - bW \equiv 0 \pmod{p^r} \qquad (3.29)$$

In this case, the conversion from p-adic code to rational form involves the solution of the above congruential relation. This is the same as solving the diophantine equation:

$$a(p^{r/2}-1) - bW = kp^r \quad , \quad k = \pm 1, \pm 2, \ldots \qquad (3.30)$$

for various values of k, and the solution which yields an order-N Farey fraction is taken.

However, it is known, [29], that an equation of the form

$$ma - nb = 1$$

has a solution $a = a_0$ and $b = b_0$ obtained by expanding m/n as a continued fraction and taking the last but one convergent which equals b_0/a_0 and, since there are many solutions to the problem, other solutions of the form:

$$a_0^{\pm tn} \quad , \quad b_0^{\pm tm} \quad , \quad \text{for } t = 0,1,2,\ldots$$

have to be tried.

Alternatively, a faster procedure can be applied which considers the weight, W', of the reciprocal of the p-adic number which yields a value of k such that

$$k = \frac{WW' - (p^{r/2}-1)^2}{p^r} \qquad (3.31)$$

Then, a search is made for that value of t which will give the desired order-N Farey fraction.

However, the study of this conversion method shows that it is not a practically suitable technique. This is chiefly due to the following points:

1. Given a p-adic code, it is not known apriori whether the weight attributed to it is a proper weight or a pseudo-weight. This is due to the fact that the type of initial rational has to be determined, i.e., soft or hard fraction, prior to its conversion to p-adic code.

2. In the case of hard fractions, particularly, solving the congruence (3.29) requires a great amount of computation time, even if the fast procedure described above were to be used.

In general, conversion of p-adic codes to rationals using the weight assignment technique proves to be laborious and time consuming.

3.4.2 Method of Look-up Tables

In this method, a look-up table is stored for every rational number occurring in the order-N Farey sequence. To every value is assigned the corresponding p-adic representation for the required value of the prime p (and length r, if we consider Hensel codes). A direct mapping between the fields Q_p and Q (or $\hat{Q}_p$ and Q) allows the determination of the rational number.

This technique involves the computation and storage of all the codes for every value of p (and r) under consideration and, although the mapping is fast, the table generation procedure itself is laborious if various combinations of the parameters p and r are required.

3.4.3 Method of Successive Additions

Using the modified algorithm presented in section 3.3.5.2 for the p-adic additions, it is possible to successively add a p-adic code until all the last r/2 digits assume the value 0 or (p-1), therefore reaching the configuration of a positive or negative integer respectively.

Therefore, the numerator of the corresponding rational fraction is the weighted power sum of the leading r/2 digits and the denominator is the number of additions performed to reach this configuration + 1.

If the last r/2 digits equal 0, then the fraction is positive. Conversely, if they are equal to (p-1), the fraction is negative.

This method proves to be reasonably fast and extremely flexible since the conversion is systematically performed over any required value of the prime p. The ambiguity pertaining to the weight/pseudo-weight determination in the congruential technique and the complexity in generating look-up tables are thus avoided.

This technique is the one used in constructing our cryptographic scheme and throughout our computations.

3.5 Limitations of the Use of Segmented p-adic Number Systems

In this chapter we presented an analysis of finite p-adic number systems. Arithmetic operations in $\hat{Q}_p$ were considered. From this analysis we draw two main conclusions regarding the use of p-adic number systems in $\hat{Q}_p$:

1. In section 3.1 it was shown that, in order to have a unique Hensel code, the rational number should have its numerator and denominator bounded by (3.4). This presents a serious limitation of these finite systems: closure of arithmetic operations is not guaranteed since the mapping of the result from $\hat{Q}_p$ to Q may not correspond to the order-N Farey sequence.

2. The limitations of Krishnamurthy's algorithms for performing finite p-adic arithmetic present a drawback in the use of these algorithms, and, although we were able to modify these algorithms, a longer intermediate code was required to implement these modifications.

It is then apparent that the use of finite p-adic number systems confined to a fixed length r involves a constant overflow trace and a constant analysis of the structure of the operands prior to any finite p-adic operations.

For this reason and for other reasons which will be explained in chapters 5 and 6, we will consider the use of variable-length p-adic number systems for the implementation of our proposed cryptosystem.

CHAPTER 4

MATHEMATICAL THEORY OF CRYPTOGRAPHIC SYSTEMS

4.1 Introduction to Cryptosystems

The main task of a cipher system is to map a plaintext message onto a ciphertext message to be transmitted over an insecure channel. This mapping should only be known to the source and receiver, such that any eavesdropper on the insecure channel cannot decipher the ciphertext to recover the original message.

Thus, the first aim of any cryptosystem is privacy. The second important goal is authentication, i.e., the prevention of an unauthorized party to inject a message into a public channel and thus assuring the receiver of a message of the legitimacy of its sender, [13] and [14].

In 1949, Shannon discussed the theory of secrecy systems through an information theoretical approach, [49]. He laid down five criteria for unconditional secrecy, that is, the ability of a system to resist any attacks by a cryptanalyst who is given unlimited time and computational power. For a detailed discussion and evaluation of those criteria, the reader is referred to Beker and Piper's paper [8].

Since then, considerable advance has been achieved in the area of cryptography, mainly due to technological advances and to theoretical developments in information theory and computer science. As a consequence, new disciplines have emerged, such as complexity theory and the analysis of algorithms, [28] and [15].

Since unconditional security is practically impossible to achieve (the only unconditionally secure system known is the one time pad, [24], which requires an extremely large key size), researchers are striving to design increasingly computationally secure cryptosystems. By computationally secure systems, we mean systems which are not vulnerable to cryptanalytic attacks subject to limitations in the cost and time of cryptanalysis, but which would be overpowered given unlimited conditions.

As a result, conventional cryptographic systems developed from the basic monoalphabetic ciphers, through the more recent mechanical cipher devices and shift registers, [7], [8] and [10], to what is now called public-key cryptosystems. A detailed discussion of public-key systems will be presented in section 4.4.

In a conventional cryptographic system (Fig. 4.1), however, the source and receiver both share the same key. The source first enciphers the message M, based on the key k, to produce the ciphertext C:

$$C = E_k(M) \qquad (4.1)$$

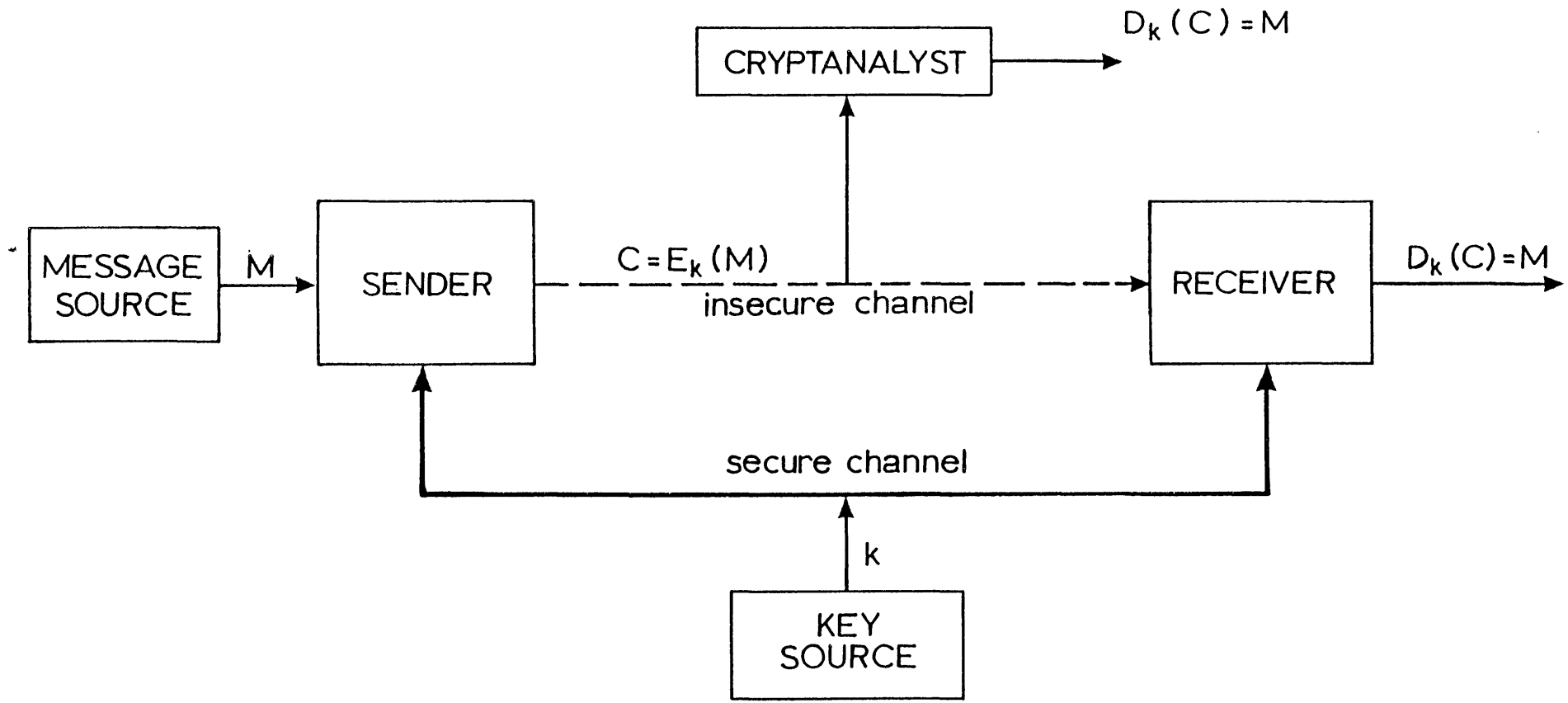


Fig. 4.1: Block Diagram of a Conventional Cryptosystem

At the receiving end, the recipient uses the same key k, but with a deciphering procedure which is the inverse of the enciphering operation, and recovers the message M:

$$D_k(C) = E_k^{-1}(E_k(M)) = M \qquad (4.2)$$

The insecure channel in Fig. 4.1 is shown by the broken lines. A cryptanalyst with a knowledge of the key k can break into the system and decipher any transmitted message. For this reason, the key has to be securely transmitted between the legitimate users. This is usually done by some physical means, thus imposing unrealistic cost and delay problems on the system.

In the next section we discuss the important feature of authentication and digital signature requirement in cryptographic systems. Then, in section 4.3, we briefly introduce the newly developed science of complexity theory which leads to a presentation of the theory of NP-completeness of public-key cryptosystems and present the major schemes based on this concept. In order to show the potential viability of each of these schemes, it will be followed by some of the attacks it has been subjected to.

4.2 Authentication and Digital Signature

In section 4.1, we briefly mentioned that one of the main goals of a cryptosystem is to provide authentication. In fact, what we mentioned earlier related to message authentication, whereby preventing the unauthorized injection of a message in the public channel.

Another feature of the authentication process is that of user authentication, where the task of the system is to verify that an indiyidual is who he claims to be [13].

In some cryptographic applications, it may not be sufficient to authenticate the validity of a message or whether it has been sent or interfered in by a third party. One further requirement is to prove that:

a) the recipient has effectively received a message from the source and,

b) the recipient has not forged or modified the message in any way.

In cases of dispute, this feature provides the receiver of a message with legal proof of the identity of the sender [14] and also bears an added protection for the sender who may thus prove his disassociation from a particular message the recipient claims to have received.

This concept is that of digital signature where, on transmission channels, it is necessary to provide the equivalent of a written signature in order to settle any dispute between the sender and receiver as to what message, if any, was sent.

This process of signing a document entails the existence of a secret key known only to the sender [12]. It should not be confused with the public key which will be discussed in future sections. But, it is obvious that both the secret and public keys are interrelated in some manner.

4.3 Complexity Theory and the Theory of NP-Completeness

In section 4.1 we mentioned that the recent advances in technology, in general, and in computer science, in particular, have created new fields of research. One such field which has a direct bearing on cryptography is that of complexity theory.

Complexity theory is a collection of results in computer science which, in essence, attempts to quantify the statement, [28]:

"Problem A is harder than Problem B".

Normally, when discussing problems in this context, we only consider decision problems whose solution is either 'yes' or 'no', [7].

A decision problem is said to belong to the complexity class P if it can be computed in polynomial time using a deterministic Turing Machine. On the other hand, if a decision problem can be solved in polynomial time by a non-deterministic machine, it is said to belong to the class NP.

One subclass of the NP problems is called the NP-complete. To define this class we have to introduce a well known problem in complexity theory which is the satisfiability problem. A full analysis of this problem can be found in [15]. For any given problem in class NP, there is a polynomial time algorithm which reduces that particular problem to the satisfiability problem. If the satisfiability problem can be solved with a polynomial time algorithm, this implies that every problem in NP is also in P. Conversely, if any problem in NP is intractable, then the satisfiability problem itself also must be intractable. Problems which share this property are the NP-complete problems.

In Fig. 4.2, we show the containment relationships between the classes P, NP and NP-complete. It is clear that any problem in class P is automatically in class NP. It is also noted that not all problems in NP and which do not belong to P are NP-complete.

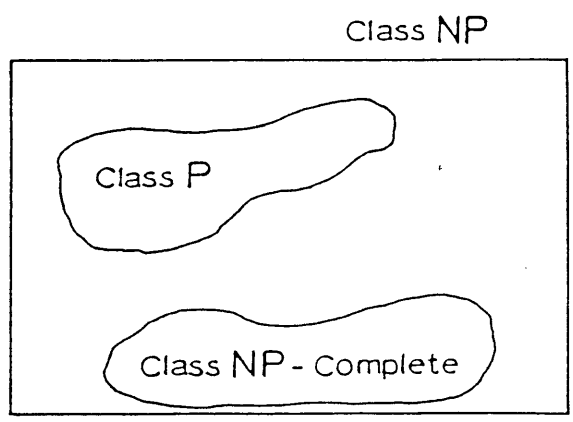


Fig. 4.2: Containment Relationships Between Complexity Classes P, NP and NP-Complete

Cryptography can draw directly from complexity theory, and particularly from the theory of NP-completeness, by attempting to provide algorithms which make any cryptanalytic task intractable.

The different systems which are presented in the following section reflect this requirement, and so does the scheme based on p-adic number systems which we are proposing in chapter 5.

4.4 Public-Key Cryptology

In section 4.1 we explained the flow of information in a conventional cipher system. The main disadvantage of such a system is the need for a physically secure channel to transmit the enciphering/deciphering key from the sender to the receiver. This same channel could not itself be used to transmit the message for reasons of capacity and delay.

This situation creates a problem of key management. Not only do the sender and receiver have to wait while the keys are sent, but also, in a system with, say, one million users, there are almost 500 billion possible connections, [14], and the cost of transmitting these keys becomes prohibitive. There is also one added risk which is that of the "extent of security" of the presumed secure channel.

These problems have generated the idea of public-key systems and, although Diffie and Hellman, [14], classify such systems into two categories, that of public-key distribution (as the Diffie-Hellman

scheme itself [14] and the Merkle scheme [36]) and of public-key cryptosystem, we will follow Beker and Piper's nomenclature, [8], and group them under the same heading of public-key cryptosystems.

Such systems are based on the concept that, although a public entry resides in a directory and the enciphering and deciphering algorithms are also made public, a cryptanalyst cannot decipher a message since this operation requires the receiver's own secret key to be exerted upon the deciphering algorithm.

As indicated in Fig. 4.3, the sender also uses his own secret key to encipher the message. The system then becomes a two-way communication system whee a cryptanalyst's task is made as hard as attempting to solve an NP problem: although he can access the public directory and use the public keys deposited, he still necessarily needs the secret key of the receiver in order to be able to decipher the encrypted message.

Three of the most realistic public-key cryptosystems which we shall discuss in detail over the next section are the Merkle-Hellman, Diffie-Hellman and Rivest-Shamir-Adleman systems.

4.4.1 The Diffie-Hellman System

The Diffie-Hellman key distribution scheme [13] makes use of the apparent difficulty of computing discrete logarithms over a finite Galois field GF(p) where p is a prime.

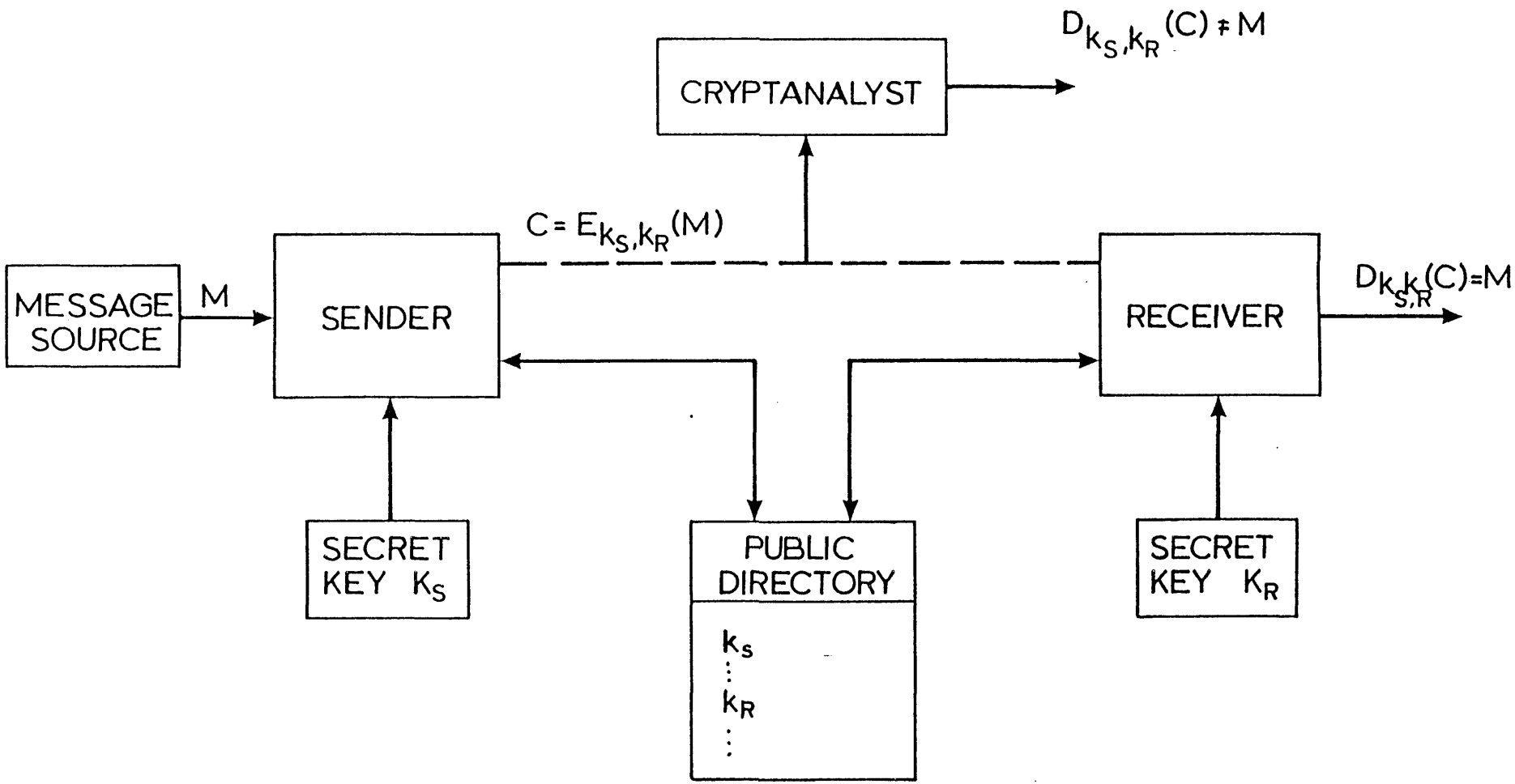


Fig. 4.3: Block Diagram of a Public-Key Cryptosystem

If α is a fixed primitive element of p, let

$$y \equiv \alpha^x \pmod{p} \quad ; \quad 1 \leq x \leq p-1 \tag{4.3}$$

In this case x is referred to as the logarithm of y to the base α modulo p:

$$x \equiv \log_\alpha y \pmod{p} \; ; \quad 1 \leq y \leq p-1 \tag{4.4}$$

or, alternatively, as the index of y to the base α [5]:

$$x = \text{ind}_\alpha y \tag{4.5}$$

In this system, each user generates an independent random integer, x_A (for user A), chosen uniformly from the set $\{1,2,\ldots,p-1\}$. x_A is a secret integer known only to A. But, whereas x_A is secret, the value y_A is placed in a public directory, such that,

$$y_A \equiv \alpha^{x_A} \pmod{p} \tag{4.6}$$

Wherever two users, A and B, want to communicate privately, they use the key:

$$K_{AB} \equiv \alpha^{x_A x_B} \pmod{p} \tag{4.7}$$

User A computes K_{AB} by obtaining y_B from the public directory and letting:

$$K_{AB} \equiv y_B^{X_A} \pmod{p} \tag{4.8}$$

$$\equiv (\alpha^{x_B})^{X_A} \pmod{p} \tag{4.9}$$

$$\equiv \alpha^{x_A x_B} \pmod{p} \tag{4.10}$$

In the same way, user B obtains K_{AB} through:

$$K_{AB} \equiv y_A^{X_B} \pmod{p} \tag{4.11}$$

In order to decipher any message, a cryptanalyst will have to compute the common secret key K_{AB} such that:

$$K_{AB} \equiv y_A^{(\log_\alpha y_B)} \pmod{p} \tag{4.12}$$

$$\equiv y_B^{(\log_\alpha y_A)} \pmod{p} \tag{4.13}$$

4.4.2 Comments on the Diffie-Hellman System

As mentioned in section 4.4.1, the Diffie-Hellman algorithm relies on the complexity of computing discrete logarithms over GF(p). It is quite simple to compute y from x in (4.3). This operation takes at most $O(2\log_2 p)$ multiplications [25].

On the other hand, computing x from y is much more complicated. One of the best algorithms which computes indices modulo p runs in exponential time $O(p^{1/2})$ [26]. Another algorithm which runs in subexponential time, i.e., which has a running time better than $O(p^{\varepsilon})$ for all $\varepsilon > 0$, was developed by Adleman [2] and requires $O(e^{\sqrt{\log_e(p)\log_e \log_e(p)}})$.

However, it has been shown [44], [45] that if (p-1) has at least one large prime factor or if p is large [2] such that p > 200 bits, then indications show that exponentiation modulo p becomes a one-way function and would then fall in class NP.

The fundamental criticism to be directed to this scheme is that, due to the fact that both entries y_A and y_B reside "permanently" in the public file, there is great concern that the repeated use of the same key might severely compromise it and hence the encipherement protocol itself may be compromised.

In the p-adic-based cryptosystem that we propose in chapter 5 we show how to overcome this fundamental shortcoming of the Diffie-Hellman algorithm.

4.4.3 The Merkle-Hellman System

Merkle and Hellman [38] have devised a cryptosystem based on the trapdoor knapsack problem which is known to be an NP-complete problem.

Generally, a trapdoor one-way function is an easily calculated function for which it is computationally infeasible to compute the inverse function unless certain specific information which was initially used in the design of the function is known.

Given a vector of integers, a:

$$a = (a_1, a_2, \ldots, a_n) \tag{4.14}$$

and an integer c, the knapsack problem consists of finding a subset of the $\{a_i\}$ such that the sum of elements of the subset is equal to c. In other words, it is required to find a binary vector $\mathbf{x}$ of n elements, such that:

$$c = a \cdot x$$

$$= \sum_{i=1}^{n} a_i x_i \tag{4.15}$$

To formulate their cryptographic algorithm, Merkle and Hellman start by considering a relatively simple knapsack problem and build it into a more complex form.

To encrypt a message, first it is divided into n-bit blocks $(x_1, x_2, \ldots, x_n)$, then the vector a is used to form the dot products, as in (4.15), thus yielding the ciphertext c.

To recover the x_i from a knowledge of c is believed to be computationally infeasible if the a_i and x_i have been chosen randomly.

To add one extra degree of complexity, when considering an effective cryptosystem, the algorithm for generating public keys first generates a simple random knapsack vector a' (with several hundred elements) which is kept secret. It also generates a random number m, such that,

$$m > \sum_{i=1}^{n} a'_i \tag{4.16}$$

and a random pair ω and ω^{-1}, given by

$$\omega\omega^{-1} \equiv 1 \pmod{m} \tag{4.17}$$

Finally, the public knapsack vector, or enciphering key, is obtained by a component multiplication:

$$a_i \equiv \omega\, a'_i \pmod{m} \; ; \quad i = 1,2,\ldots,n \tag{4.18}$$

When a user A wishes to send a message x to B, he computes the ciphertext c as in (4.15) and sends c to B. B uses his secret multiplicative inverse ω^{-1} modulo m and performs the following computation:

$$c' \equiv \omega^{-1} c \pmod{m}$$

$$\equiv \omega^{-1} \sum_{i=1}^{n} a_i x_i \pmod{m}$$

$$\equiv \omega^{-1} \sum_{i=1}^{n} \{[\omega\, a'_i \pmod{m}]\, x_i\} \pmod{m}$$

$$\equiv \sum_{i=1}^{n} \{[\omega^{-1} \omega\, a'_i \pmod m)]\, x_i\} \pmod m$$

$$\equiv \sum_{i=1}^{n} a'_i\, x_i \pmod m$$

$$\equiv a' \cdot x \qquad (4.19)$$

since $m > \sum_{i=1}^{n} a'_i$.

The vector a is placed in a public directory as a user's key while ω^{-1} and m are kept secret to decipher any message which has been enciphered with this public key.

Fig. 4.4 shows the flow of information in the Merkle-Hellman system.

4.4.4 Comments on the Merkle-Hellman System

Although known to be in NP-complete, the knapsack problem has been shown [23], [48] to be solvable under some assumptions. This fact leads to the necessity of generating specific sequences of the vector a [37].

Furthermore, from a practical point of view, the public key vector a is extremely large. Merkle and Hellman recommend the following minimum parameters of a to ensure secure communication:

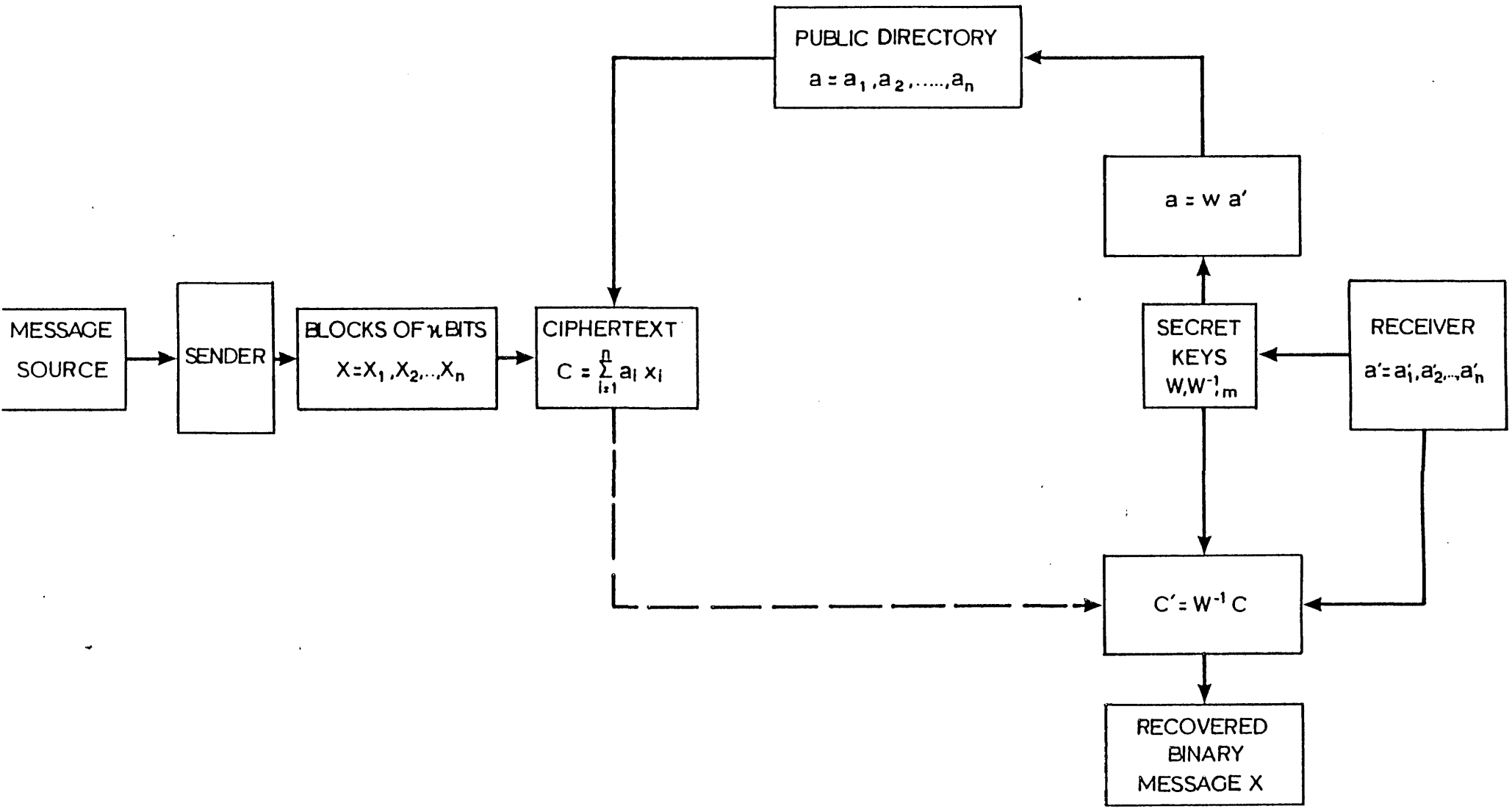


Fig. 4.4: Flow of Information in the Merkle-Hellman Cryptosystem

1. $n = 100$

2. each randomly chosen a'_i is a $(99+i)$-bit natural number.

These specifications, although they safeguard the cryptographic security of the system, the orders of magnitude involved are extremely impractical to be incorporated in a public directory which may be shared by a wide number of users.

4.4.5 The Rivest-Shamir-Adleman System

The Rivest-Shamir-Adleman [46] (RSA) system uses exponentiation in a different way from that used by Diffie and Hellman. They make use of the fact that, given the modern computer technology, it is computationally easy to find large prime numbers of the order of 100 digits. On the other hand, given the same computer power, it appears that factoring the product of two such primes is computationally infeasible and hence belonging to the class NP.

In this system, user A selects two large primes p and q at random. Then he obtains their product n:

$$n = p \cdot q \tag{4.20}$$

Consequently, since p and q are primes (refer to equation (2.19) in chapter 2), the Euler totient function $\phi(n)$ is computed as:

$$\phi(n) = (p-1)(q-1) \tag{4.21}$$

The encryption key, e, is then chosen at random from the interval [2, $\phi(n)-1$], such that

$$(e, \phi(n)) = 1 \qquad (4.22)$$

The decryption key, d, is the multiplicative inverse of e modulo $\phi(n)$:

$$ed \equiv 1 (\text{mod } \phi(n)) \qquad (4.23)$$

The pair of integers (e,n) is made public while p,q, $\phi(n)$ and d are all kept secret. A message is then represented as a sequence of integers m_1, m_2,..., such that

$$0 \leq m_i \leq n-1 \qquad (4.24)$$

and m_i is about 700 bits.

Each block m_i is then enciphered using the public keys e and n of the receiver and the ciphertext, c, is produced according to:

$$c_i \equiv m_i^e \quad (\text{mod } n) \qquad (4.25)$$

To decipher the message block m_i, c_i is raised to the receiver's secret power d:

$$c_i^d \equiv (m_i^e)^d \quad (\text{mod } n)$$

$$\equiv m_i \quad (\text{mod } n) \qquad (4.26)$$

and since $m_i < n$, the value obtained corresponds to the originally transmitted message block.

In Fig. 4.5 we describe the flow of information in the RSA system.

4.4.6 Comments on the RSA Systems

The RSA system has, so far, withstood many attacks and, unless large improvements are made in the factorisation problem or techniques for inverting m_i^e without requiring the secret decryption key d, are found, it remains potentially secure.

However, an algorithm for breaking the RSA system has been independently devised in [51]. It relies on the possibility that an encrypted message m can be decrypted by successively re-encrypting the ciphertext, c, where $c \equiv m^e \pmod n$.

More formally, this can be shown by setting c_i to c then compute

$$c_{i+1} \equiv c_i^e \pmod n \qquad (4.27)$$

Congruence (4.27) is then repeated until $c_{i+1} = c$. Then $c_i = m$.

This is illustrated in Table 4.1 where we consider the same example used by Rivest et. al. [46].

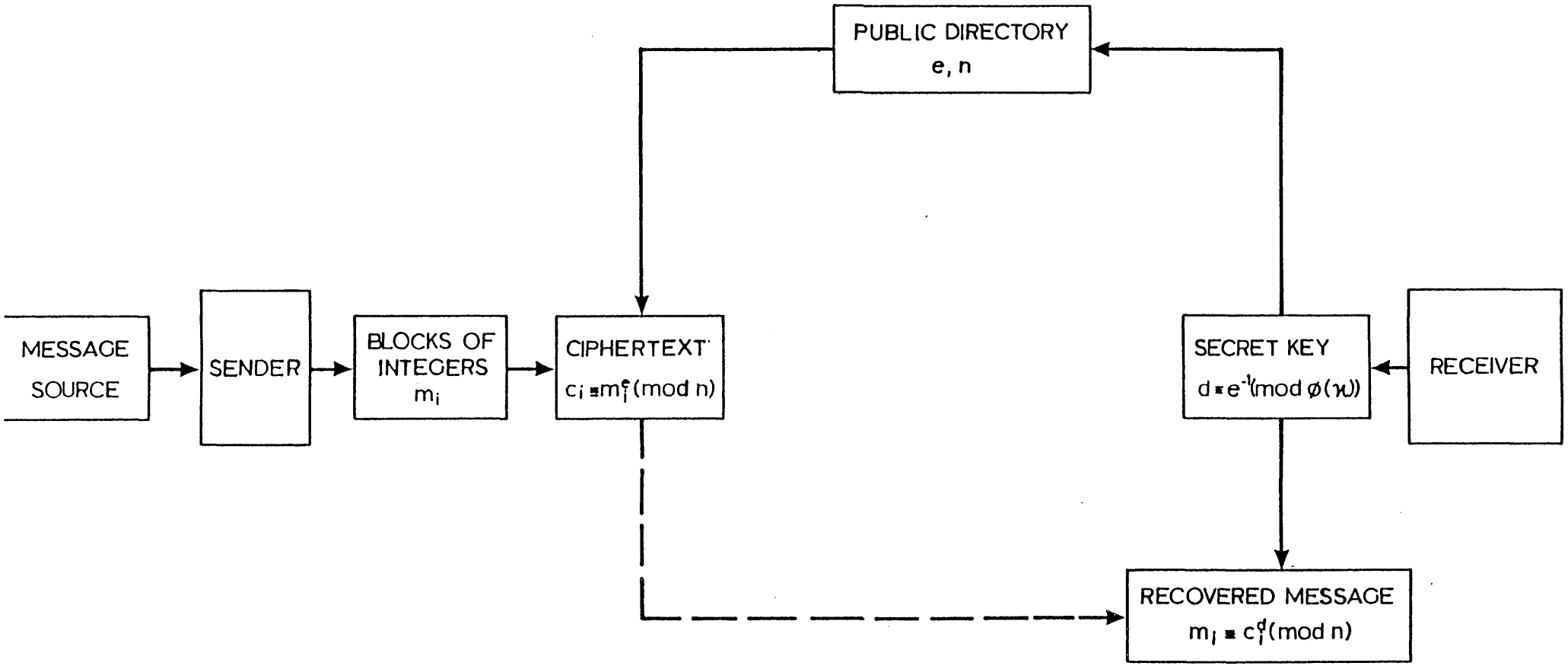


Fig. 4.5: Flow of Information in the Rivest-Shamir-Adleman (RSA) Cryptosystem

For the case p = 47 and q = 59, then n = p·q = 47.59 = 2773 and d = 157. Consequently, $\phi(2773) = 46.58 = 2668$ and the encrypting key e is the multiplicative inverse of d modulo 2668. Thus e = 17.

Rivest et. al., in their example, considered the message

ITS ALL GREEK TO ME

(Julius Caesar, I, ii, 288, paraphrased)

and they encoded two letters per block, substituting a two-digit number for each letter: blank = 00, A = 01, B = 02 ,..., Z = 26. Thus the whole message is encoded as:

0920 1900 0112 1200 0718

0505 1100 2015 0013 0500

Since e = 17, then the message has the following ciphertext:

0948 2342 1084 1444 2663

2390 0778 0774 0219 1655

In Table 4.1, we consider the first block of ciphertext where $c_1 = 948$ and by successively raising this block to e = 17 modulo 2773, it is seen that it converges back to c_1 in 44 iterations. The 43^{rd} iteration, however, is the initial encoded message, 920, corresponding to the first two letters I and T of the transmitted message.

Table 4.1: Successive iterations for breaking the RSA system for the case e = 17, n = 2773, C = 948

ITERATION	C^{17} (MOD 2773)
	948
1	1207
2	723
3	802
4	2305
5	2151
6	2552
7	1156
8	1243
9	2033
10	900
11	1510
12	2187
13	1679
14	251
15	212
16	299
17	2387
18	841
19	2218
20	4
21	27
22	1136
23	2100
24	535
25	2682
26	2493
27	271
28	2364
29	263
30	1431
31	153
32	712
33	617
34	2375
35	2572
36	63
37	2092
38	487
39	507
40	653
41	1325
42	192
43	920
44	948

Another more complicated algorithm which also attempts to break the RSA cryptosystem was developed by Herlestam [23]. Although both algorithms have a small probability of success since they depend on particular values of e and $\phi(n)$, they offer some indications regarding a higher probability of breaking the system.

It should be emphasized, however, that these algorithms do not address the factorisation problem in any way but they rely on the ciphertext itself and the public information in order to recover the corresponding plaintext.

CHAPTER 5

A NEW PUBLIC-KEY CRYPTOGRAPHIC SCHEME BASED ON p-ADIC NUMBER SYSTEMS

5.1 Introduction to the Proposed System

In chapter 4 we described the mathematical theory of public-key cryptosystems. We showed that all the algorithms developed for this class of security systems rely on number theoretic concepts.

In this chapter, we extend this principle by devising a new algorithm for public-key encryption. This algorithm is an extension of both, the Diffie-Hellman and RSA systems. It exploits the features of each system's one-way functions: in the case of the Diffie-Hellman system, the difficulty of computing discrete logarithms over finite fields and, in the case of the RSA system, the complexity of factoring a known integer into its prime factors.

Furthermore, the scheme makes use of the p-adic number system discussed in chapters 2 and 3. It is demonstrated that, by using this system to encipher a given alphabet, any cryptanalytic approach will have to consider a very large search space to determine the prime p which allows the decryption of the message to be performed.

To achieve this, variable-length p-adic number systems will be used. The structure of the corresponding p-adic codes will be given in section 5.2 and, in section 5.3, we discuss the problem of selecting the alphabet required for the proposed cryptosystem.

The encryption and decryption algorithms are presented in section 5.4.

In section 5.5, a further extension of the p-adic system is proposed, where the algorithm is performed over a g-adic ring instead of a finite p-adic field. The search space then increases exponentially and any cryptanalytic attack will not only have to consider prime values but, also, any power of a prime.

The algorithms for encrypting and decrypting in this extended system will also be derived and the method of converting a g-adic sequence of integers into its corresponding p-adic sequence will be explained.

5.2 p-adic Code Structure for Cryptographic Implementation

In chapter 3, we pointed out some of the limitations of using finite-segment p-adic number systems.

To these limitations, we add one extra constraint. For implementation in a cryptographic scheme where the alphabet is made public to all users, a knowledge of the fixed finite length r of $H(p,r,\alpha)$ where α is an element of the alphabet and p is an unknown value, gives direct indications to the value of p. This is due to the inherent structure of the finite p-adic number systems and which are necessarily bounded by the bounds given in (3.4).

On the other hand, a knowledge of r is required in order to perform the conversion algorithm in any decryption process to recover the message α.

To solve this paradox, we suggest that variable-length p-adic number systems be used. This is done as follows.

Given an alphabet set and based on a particular prime value, p, the sender of a message can compute the period λ of the variable-length p-adic code according to the analysis reported in section 2.3 of chapter 2. Also, the aperiodic elements in this expansion can be computed as shown in section 2.4 of the same chapter. The algorithm given in section 2.4 also allows the first periodic element, a pointer to which will be denoted by λ_1, to be detected.

Hence, the structure shown in Fig. 5.1 is suggested for the variable-length p-adic code.

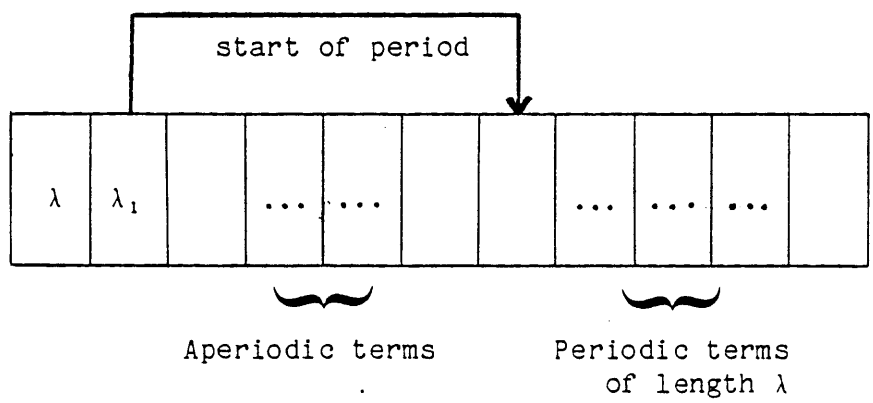


Fig. 5.1 Structure of the variable-length p-adic code based on the period λ

In Appendix B we give a table of such codes for p = 5 and for a maximum numerator/denominator value of 17, for comparison with the finite-segment Hensel codes with r = 4, given in Appendix A.

To convert such a code back to its rational value α, we use the successive addition algorithm described in section 3.4.3. An integer representation is reached when, for a particular even value of r, the last $\frac{r}{2}$ digits = 0. However, to find this even value of r, we proceed as follows.

If γ is the largest value represented in the alphabet, then set

$$\gamma = \sqrt{\frac{p^r - 1}{2}} \qquad (5.1)$$

$$\therefore \quad 2\gamma^2 = p^r - 1$$

$$\therefore \quad r = \left\lceil \frac{\log(2\gamma^2+1)}{\log p} \right\rceil \qquad (5.2)$$

If the obtained value of r is odd, then

$$r = r + 1 \qquad (5.3)$$

Consequently, a code corresponding to the variable-length p-adic code sent, but of length r, can be formed by the receiver and hence converted to its unique rational value in the alphabet.

Another algorithm, however, considers the canonic p-adic expansion of α:

$$\alpha = \sum_{n=i}^{\infty} a_n p^n \qquad (5.4)$$

Since α is periodic with periodicity λ, then we can write α in the form:

$$\alpha = a_i p^i + a_{i+1} p^{i+1} + \ldots + a_{i+k} p^{i+k}$$

$$+ b_1 p^{i+k+1} + b_2 p^{i+k+2} + \ldots + b_\lambda p^{i+k+\lambda}$$

$$+ b_1 p^{i+k+\lambda+1} + \ldots + b_\lambda p^{i+k+2\lambda}$$

$$+ \ldots \qquad (5.5)$$

$$= p^i(a_i + a_{i+1} p + \ldots + a_{i+k} p^k)$$

$$+ p^{i+k+1}(b_1 + b_2 p + \ldots + b_\lambda p^{\lambda-1})$$

$$+ p^{i+k+\lambda+1}(b_1 + b_2 p + \ldots + b_\lambda p^{\lambda-1})$$

$$+ \ldots \qquad (5.6)$$

$$= p^i A + p^{i+k+1} B(1+p^\lambda + p^{2\lambda} + \ldots) \qquad (5.7)$$

where

$$\left.\begin{aligned} A &= a_i + a_{i+1} p + \ldots + a_{i+k} p^k \\ \text{and} \\ B &= b_1 + b_2 p + \ldots + b_\lambda p^{\lambda-1} \end{aligned}\right\} \qquad (5.8)$$

Then, (5.7) equals

$$p^i A + p^{i+k+1} B \frac{1}{1-p^\lambda} \tag{5.9}$$

since, in Q_p,

$$1 + p^\lambda + p^{2\lambda} + \ldots + p^{(n-1)\lambda} = \frac{1-p^{n\lambda}}{1-p^\lambda} \tag{5.10}$$

and

$$\frac{1-p^{n\lambda}}{1-p^\lambda} \rightarrow \frac{1}{1-p^\lambda} \quad \text{as} \quad n \rightarrow \infty.$$

Thus, A is the sequence of aperiodic terms in the structure, while B is the sequence of periodic terms (notice that b_1 in the above analysis corresponds to the element pointed at by λ_1 in the code), and by computing (5.8) and (5.7), the equivalent rational value of α can be determined.

However, this algorithm is more complicated than the fast successive addition algorithm since it involves continuous exponentiation, multiplication and addition. Furthermore, the magnitude of the result of the exponentiation of the prime p can cause overflow problems since the above operations are no longer carried out in a modular field.

We thus revert to the successive addition algorithm described in this section and based on the proposed p-adic code structure to convert the codes from Q_p to Q.

5.3 Choice of Alphabet

Since our proposed cryptosystem is based on p-adic number systems, then, initially, our alphabet will consist of rational numbers.

For each character, α, in the alphabet two integer values are assigned, one for the numerator and another for the denominator of α.

There are two considerations to be made in this respect.

First, notice that in the p-adic code structure given in section 5.2, the p-adic point is not considered as part of the code. If a p-adic point were incorporated, then it is possible to deduce the value of p. Consequently, radix fractions corresponding to p must be omitted from the alphabet.

This is easily done since, once the alphabet has been generated and stored for public use, users of the system will select values of p such that

$$p > \Upsilon \qquad (5.11)$$

where Υ is the largest value represented in the alphabet set.

In a typical cryptographic system, this is guaranteed because it is necessary to use large values of p (of the order of 200 digits).,

The second consideration relates to what we label "fractional ambiguity". By fractional ambiguity we mean fractions $\frac{c}{d}$ such that if $(c,d) > 1$ they can be reduced to a fraction $\frac{a}{b}$ where $(a,b) = 1$ and which already exits in the alphabet.

It can be ensured that this situation does not occur if the elements of an alphabet are drawn from an order-N Farey sequence.

5.4 Realization of the Scheme Based on p-adic Fields

In Fig. 5.2 we show the flow of information in the proposed cryptographic system based on p-adic number systems. The scheme involves two prime numbers p and p_{AB}. These prime numbers are chosen randomly and, as in any cryptographic scheme based on discrete logarithms, p must be chosen such that p-1 has at least one large prime factor.

The system is best described by explaining the corresponding encryption and decryption algorithms.

5.4.1 The p-adic Encryption Algorithm

Suppose that A and B want to secretly communicate together, then in this system, as in the Diffie-Hellman system, we propose that p and its primitive element α be universal to all users of the system along with the afore-mentioned alphabet.

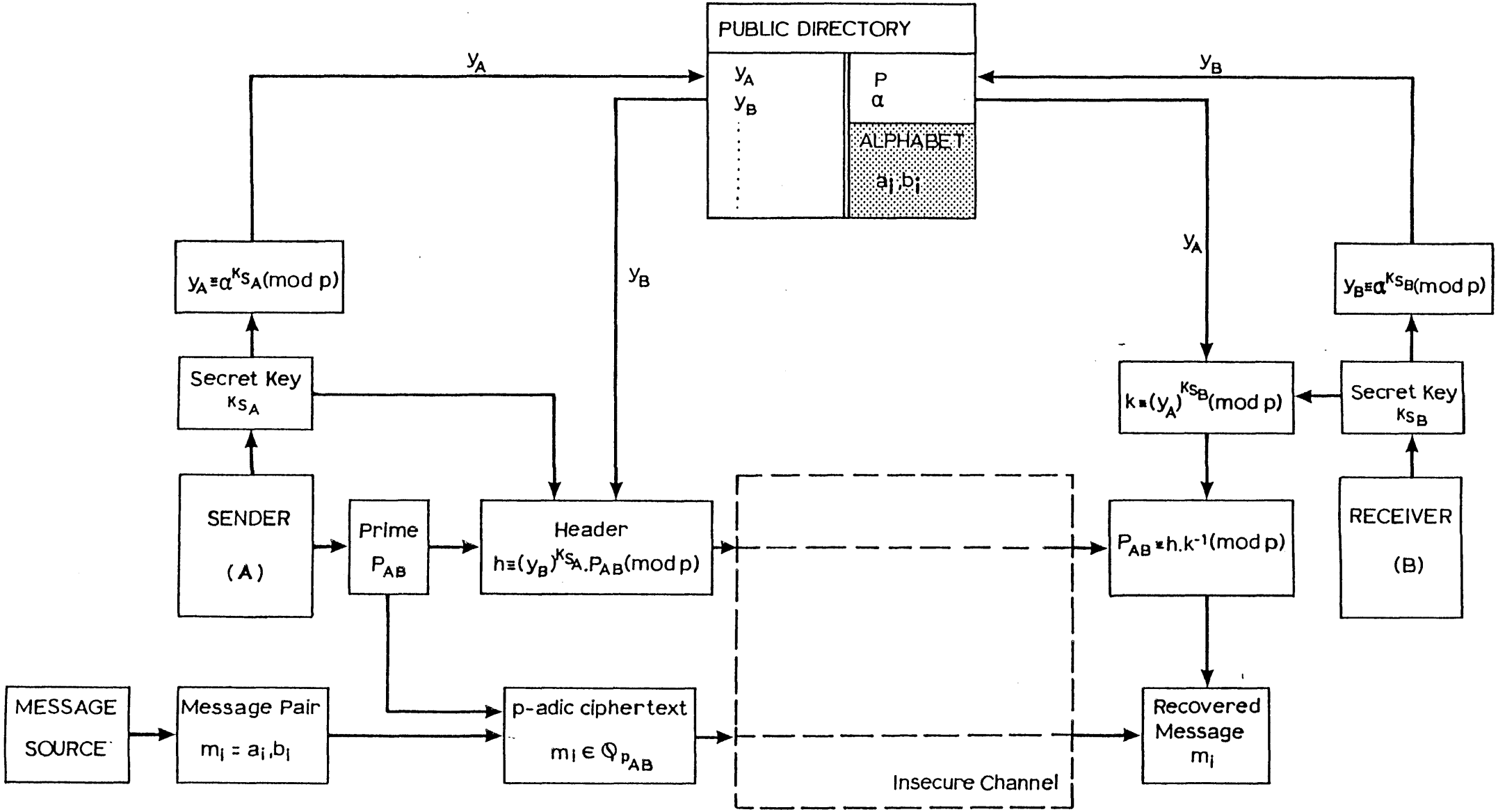


Fig. 5.2: Flow of Information in the p-adic-Based Cryptosystem

A chooses a secret key, K_{s_A} and places y_A in a public directory, such that:

$$y_A \equiv \alpha^{K_{s_A}} \pmod{p} \tag{5.12}$$

and so does B, with his secret key K_{s_B}:

$$y_B \equiv \alpha^{K_{s_B}} \pmod{p} \tag{5.13}$$

Then, if A wants to send a message, m, to B, he first selects a prime p_{AB}, such that,

$$p_{AB} < p \tag{5.14}$$

This prime constitutes the basis of the p-adic code in the ciphertext (and which, actually, corresponds to p of Q_p in chapter 2). p_{AB} is a hidden information which will be sent to B as the header, h, prior to the sequence of p-adic codes of the message m. This is done as follows:

$$h \equiv (y_B)^{K_{s_A}} \cdot p_{AB} \pmod{p} \tag{5.15}$$

since y_B is in the public directory and K_{s_A} is known only to A. Then, the ciphertext is sent over the insecure channel and is composed of the variable-length p-adic code of m.

5.4.2 The p-adic Decryption Algorithm

At the receiving end, B who is anticipating to receive a message from A, first acquires the header h. This header contains the

necessary information p_{AB} which allows B to decipher the subsequent code. To "filter" p_{AB} from the header h, B performs the following computation:

$$\begin{aligned} h &\equiv (y_B)^{K_{s_A}} \cdot P_{AB} \pmod{p} \\ &\equiv (\alpha^{K_{s_B}})^{K_{s_A}} \cdot p_{AB} \pmod{p} \\ &\equiv (\alpha^{K_{s_A}})^{K_{s_B}} \cdot p_{AB} \pmod{p} \end{aligned} \tag{5.16}$$

But,

$$\alpha^{K_{s_A}} \pmod{p} \equiv y_A$$

which is in the public directory, and K_{s_B} is only known to B. Hence, the value

$$k \equiv (\alpha^{K_{s_A}})^{K_{s_B}} \pmod{p} \tag{5.17}$$

can be computed.

Since p is a prime, then $k^{-1}(p)$ exists. By calculating this value of $k^{-1}(p)$, B recovers p_{AB} such that,

$$p_{AB} \equiv h \cdot k^{-1} \pmod{p} \tag{5.18}$$

and consequently B is able to decipher the encoded p-adic message based on this value of p_{AB} and according to the method reported in section 5.2.

5.5 Realization of the Scheme Based on g-adic Rings

In section 5.4 we described the suggested cryptographic algorithm based on a prime p_{AB} and its corresponding p-adic field $Q_{p_{AB}}$.

In this section, we extend this concept even further with the view of making any cryptanalyst's task computationally infeasible. For this purpose, we suggest the implementation of the scheme over the g-adic ring Q_g. In fact, this is a generalization of the p-adic case.

For a formal introduction to g-adic rings, we refer the reader to Mahler's book [35]; in this thesis we are concerned with the application of such rings in our cryptographic scheme. This, again, will be explained in more detail through the following encryption and decryption algorithms.

5.5.1 The g-adic Encryption Algorithm

As mentioned earlier, this is an extension of the previously detailed algorithm for the case $GF(p_{AB})$.

In the g-adic case, if A wants to send a message, m, to B, then the ciphertext is taken over p_{AB}^n where $n \in Z^+$ (and $n < p$, the universal prime in the system).

A, then, transmits the first header, h_1, as in (5.15):

$$h_1 \equiv (y_B)^{K_{s_A}} \cdot p_{AB} \pmod{p} \qquad (5.19)$$

and also sends a second header, h_2, such that

$$h_2 \equiv (y_B)^{K_{s_A}} \cdot n \pmod{p} \tag{5.20}$$

Then, he sends the g-adic sequence of integers representing m and corresponding to Q_g where

$$g = p_{AB}^n \tag{5.21}$$

The computation of the g-adic sequence in Q_g is similar to the variable-length p-adic sequence in $Q_{p_{AB}}$ with the difference that the modulus is the prime p_{AB} raised to the power n.

5.5.2 The g-adic Decryption Algorithm: The Decomposition of Q_g into Q_p

To be able to decipher the received ciphertext, based on the g-adic ring Q_g, the receiver, B, first has to recover the values of p_{AB} and n from the headers h_1 and h_2, respectively. This is done in the same manner described in section 5.4.2, namely:

$$h_1 \equiv k \cdot p_{AB} \pmod{p} \tag{5.22}$$

and

$$h_2 \equiv k \cdot n \pmod{p} \tag{5.23}$$

where k is given by congruence (5.17).

Hence,

$$p_{AB} \equiv h_1 \cdot k^{-1} \pmod{p} \tag{5.24}$$

and, similarly,

$$n \equiv h_2 \cdot k^{-1} \pmod{p} \tag{5.25}$$

Subsequently, based on the knowledge of p_{AB} and n, B can decipher the received sequence of g-adic numbers. This is done in two stages. First, the g-adic sequence is transformed into a p-adic sequence and, then, the usual p-adic to rational conversion, described in earlier sections, is performed.

The conversion from the g-adic code to the p-adic code is based on the following development, where p_{AB} is replaced by p for simplicity of notation (but which should not be confused with the universal prime of the cryptosystem).

Since we are considering the case of $g = p^n$ where $n \geq 2$, it can be shown [35] that any p^n-adic number, β, can be expressed as a p-adic number. Assume

$$\beta = \sum_{i=-j}^{\infty} \beta_i \, (p^n)^i \tag{5.26}$$

and which represents the canonic series for β, where $j \in Z$ and the coefficients β_i are p^n-adic digits $0,1,2,\ldots,p^n-1$.

To the basis, p, these coefficients can be written as:

$$\beta_i = \sum_{k=0}^{n-1} b_{in+k} \cdot p^k \qquad (5.27)$$

where $i = -j,-j+1,-j+2,\ldots$

In (5.27), the new coefficients b_{in+k} are p-adic digits, $0,1,2,\ldots,p-1$; hence, β can be expressed as the p-adic number

$$\beta = \sum_{m=-jn}^{\infty} b_m \, p^m \qquad (5.28)$$

Consequently, B, upon receiving the sequence of p^n-adic numbers, undergoes a decomposition of these numbers into the corresponding p-adic field and, finally, converts the obtained p-adic code into its rational equivalent, thereby recovering the original message sent by A.

CHAPTER 6

EVALUATION OF THE PROPOSED SYSTEM

6.1 Cryptanalytic Approaches to Breaking the p-adic-Based System

In the preceding chapter, a new cryptographic scheme based on p-adic number systems was presented. This algorithm relied on the discrete logarithm problem. Breaking the system is equivalent to breaking the Diffie-Hellman distribution scheme. However, an improvement on the Diffie-Hellman algorithm was introduced through the use of variable-length p-adic codes.

In this chapter, it is intended to subject the proposed scheme to different attacks and, through these attacks, attempt to evaluate its cryptographic viability.

To study the security of the system, the following analysis will be based on the fact that solving discrete logarithms, where the modulus p is large (> 200 bits) or if p-1 has at least one large prime factor, it is computationally infeasible to calculate the secret keys K_{s_A} and K_{s_B} from a knowledge of y_A and y_B (in congruences (5.12) and (5.13) in chapter 5) and which reside in the public file.

Consequently, the next step in the cryptanalytic attack is to aim to recover p_{AB} from a knowledge of the header h. To do this, h should be properly factorized.

Again, the factorization problem appears to be an NP problem, since factoring a number seems to be much more difficult than determining whether it is prime or composite [46]. Since this factorization problem is the basis of the RSA system which is known to be reasonably secure, one extra degree of complexity is even claimed in our scheme over the RSA system.

Whereas in this latter algorithm the problem resides in the ability to factorize n, where

$$n = p \cdot q$$

into the two distinct prime factors p and q, the proposed system, on the other hand, involves the congruence

$$h \equiv k \cdot p_{AB} \pmod{p} \qquad (6.1)$$

where there is no guarantee that k is a prime number. Hence, the modular factorization of h into a prime and a composite number is much more complicated than the decomposition into two primes.

Assuming that a cryptanalyst has available the currently non-existing ability to factorize a 200-digit long number into 2 primes, it is then safe to ascertain that it is unrealistic, given a margin of safety against future developments, to assume that such a cryptanalyst can solve the even more complex problem of factorizing the 200-digit long number into a prime and composite factors, and then selecting the correct prime p_{AB} out of all the involved possibilities.

Secondly, the possibility of breaking the system based on a ciphertext-only analysis is considered. This, again, appears to be a computationally infeasible task due to the pseudo-randomness in the distribution of the p-adic numbers in $Q_{p_{AB}}$.

The plots given in Figs. 6.1, 6.2 and 6.3 reflect this evidence. In these plots the prime $p_{AB} = 2909$ is considered and a random alphabet of 26 characters, with $\gamma = 20$, is generated. Then, message lengths of $M = 100$, 1000 and 10000 characters were randomly simulated and the frequencies of occurrence of the p-adic numbers $0,1,\ldots,2908$ are respectively plotted for each message length. It is seen from these plots that the p-adic numbers have a random distribution over the considered range.

However, one argument which is further investigated, claims that, although the p-adic numbers are randomly distributed, some of the numbers in $Q_{p_{AB}}$ do not appear in the transmitted p-adic code. By detecting such numbers, a cryptanalyst may have an indication, however extremely small, to the prime p_{AB} and hence the ability to break the system.

To confirm that this argument has no chance of breaking the proposed scheme, entropy calculation of the p-adic numbers for different primes p_{AB} was then tested. This is shown in the plots of Figs. 6.4, 6.5 and 6.6 where values of p_{AB} in the range between 0 and 10000 were considered. By simulating random messages of lengths

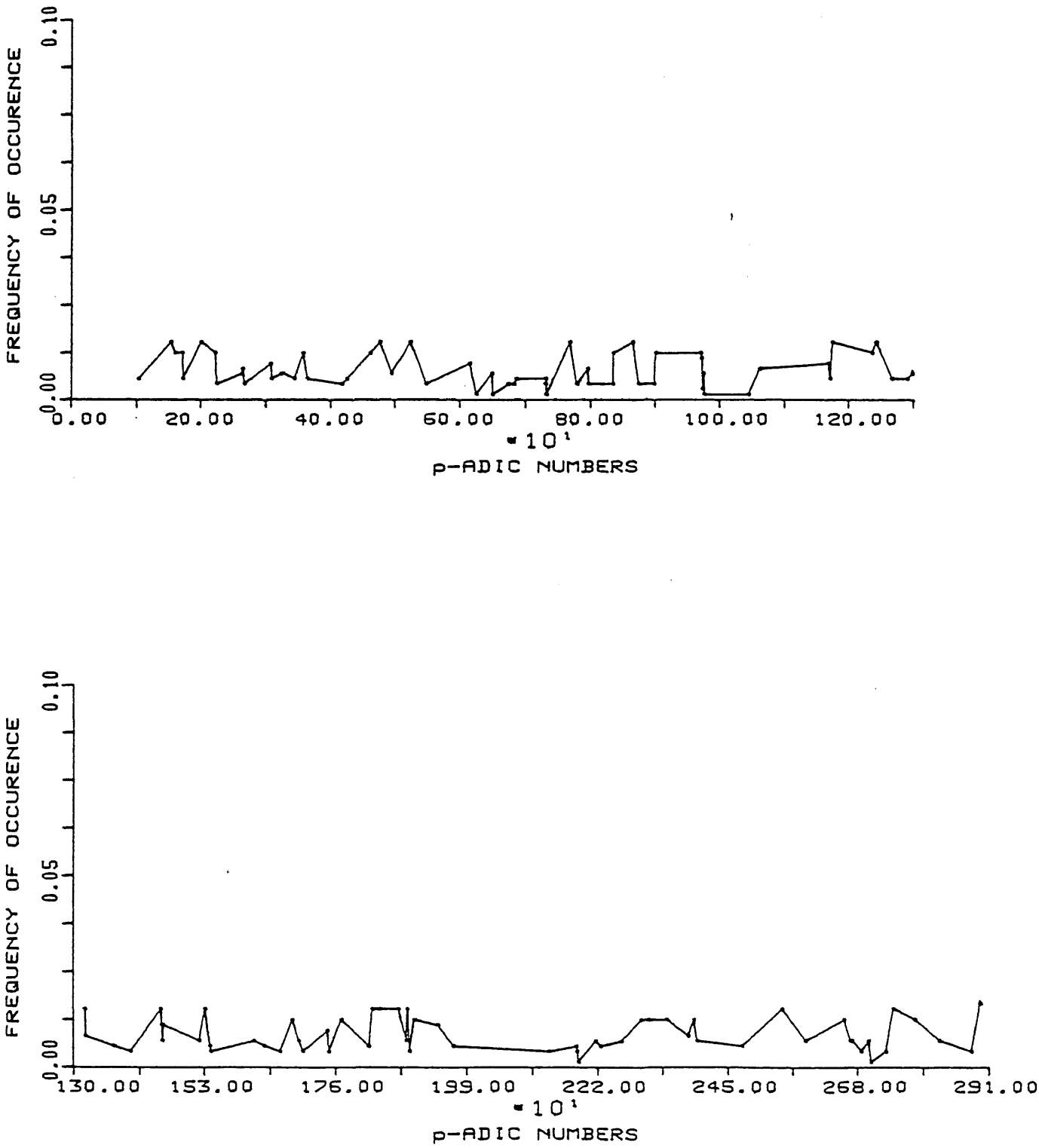


Fig.6.1 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH EQUIPROBABLE CHARACTER FREQS.

P_{AB} = 2909, M = 100

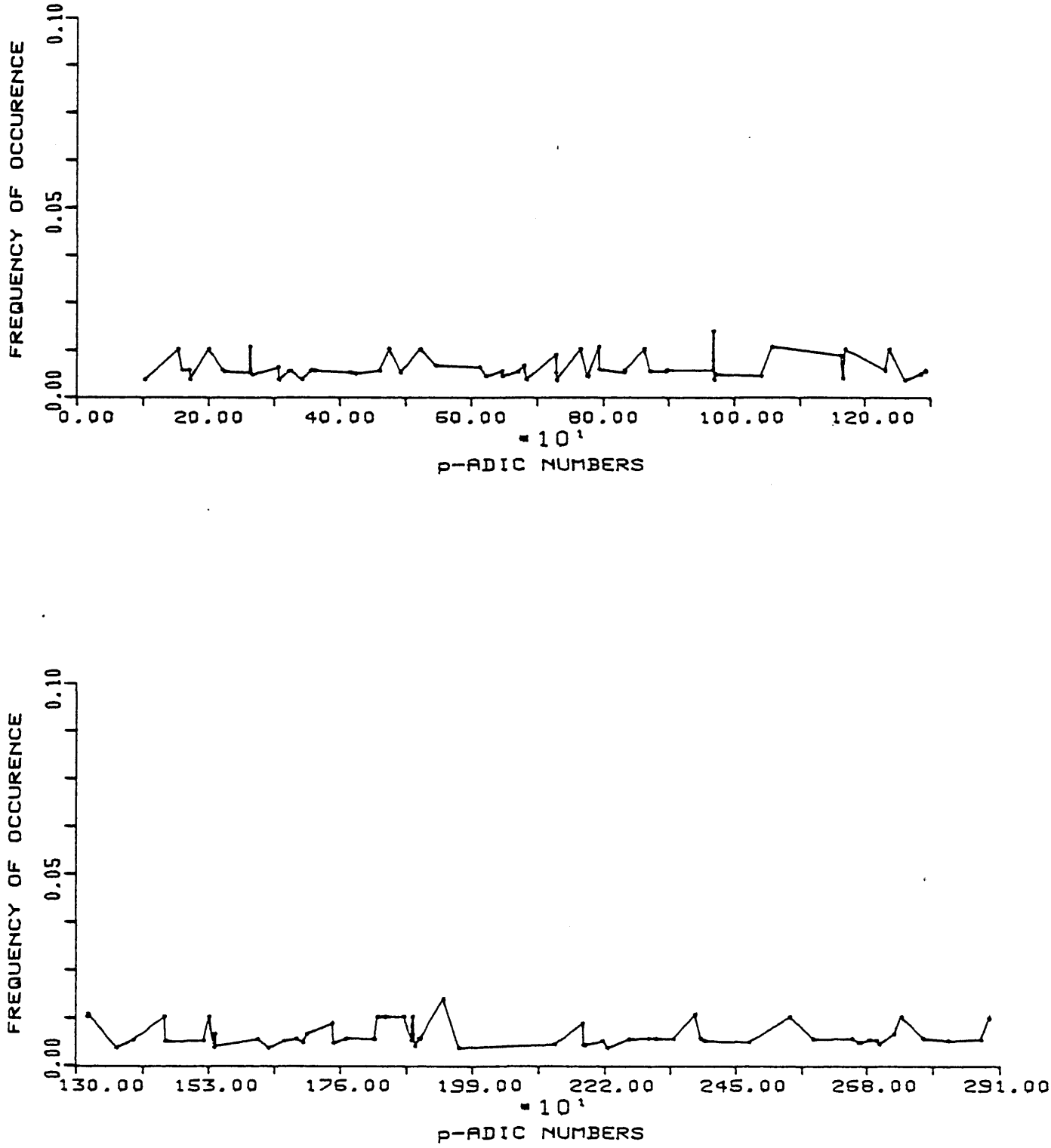


Fig.6.2 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH EQUIPROBABLE CHARACTER FREQS.

p_{AB} = 2909, M = 1000

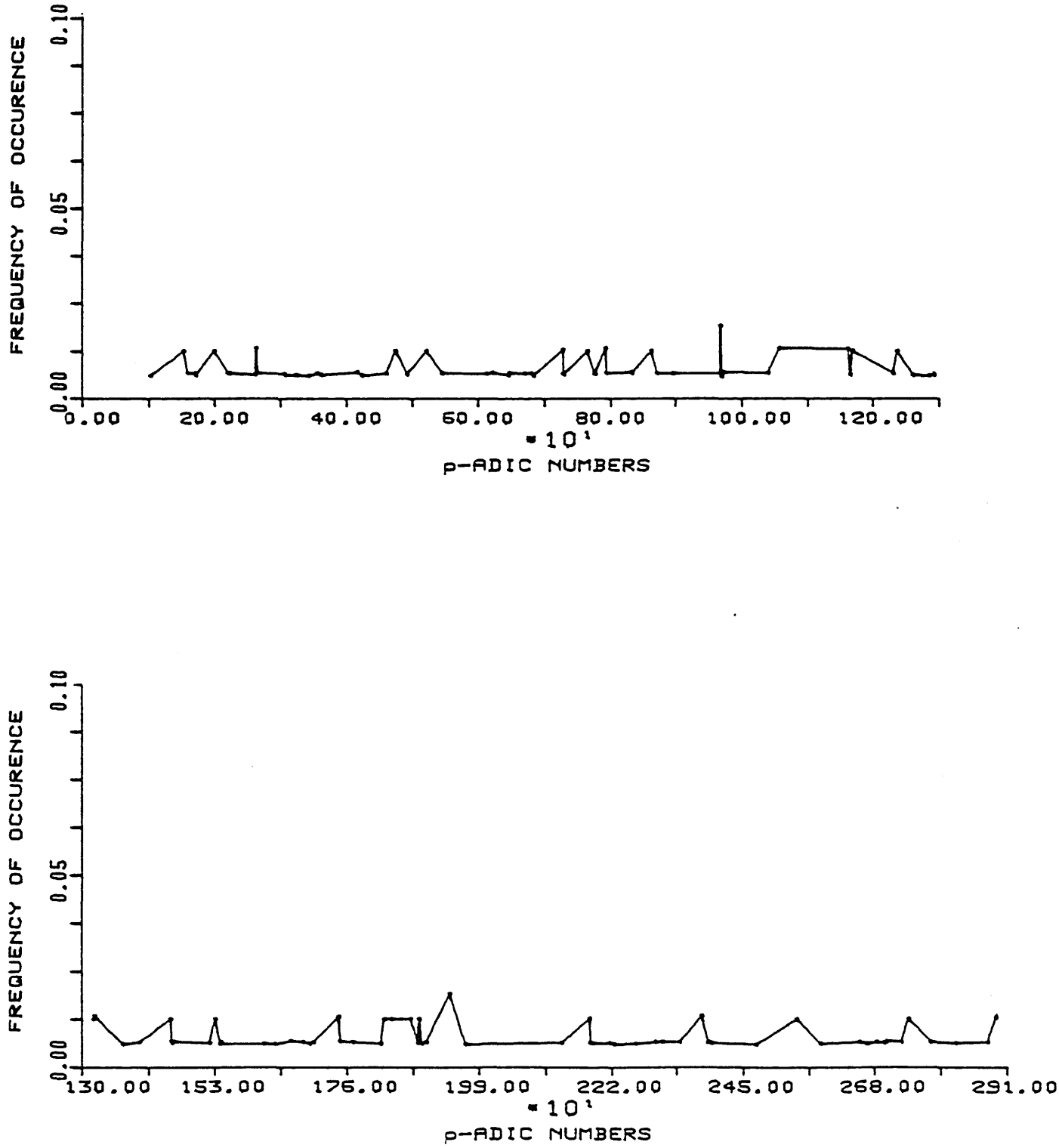


Fig.6.3 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH EQUIPROBABLE CHARACTER FREQS.

P_{AB} = 2909, M = 10000

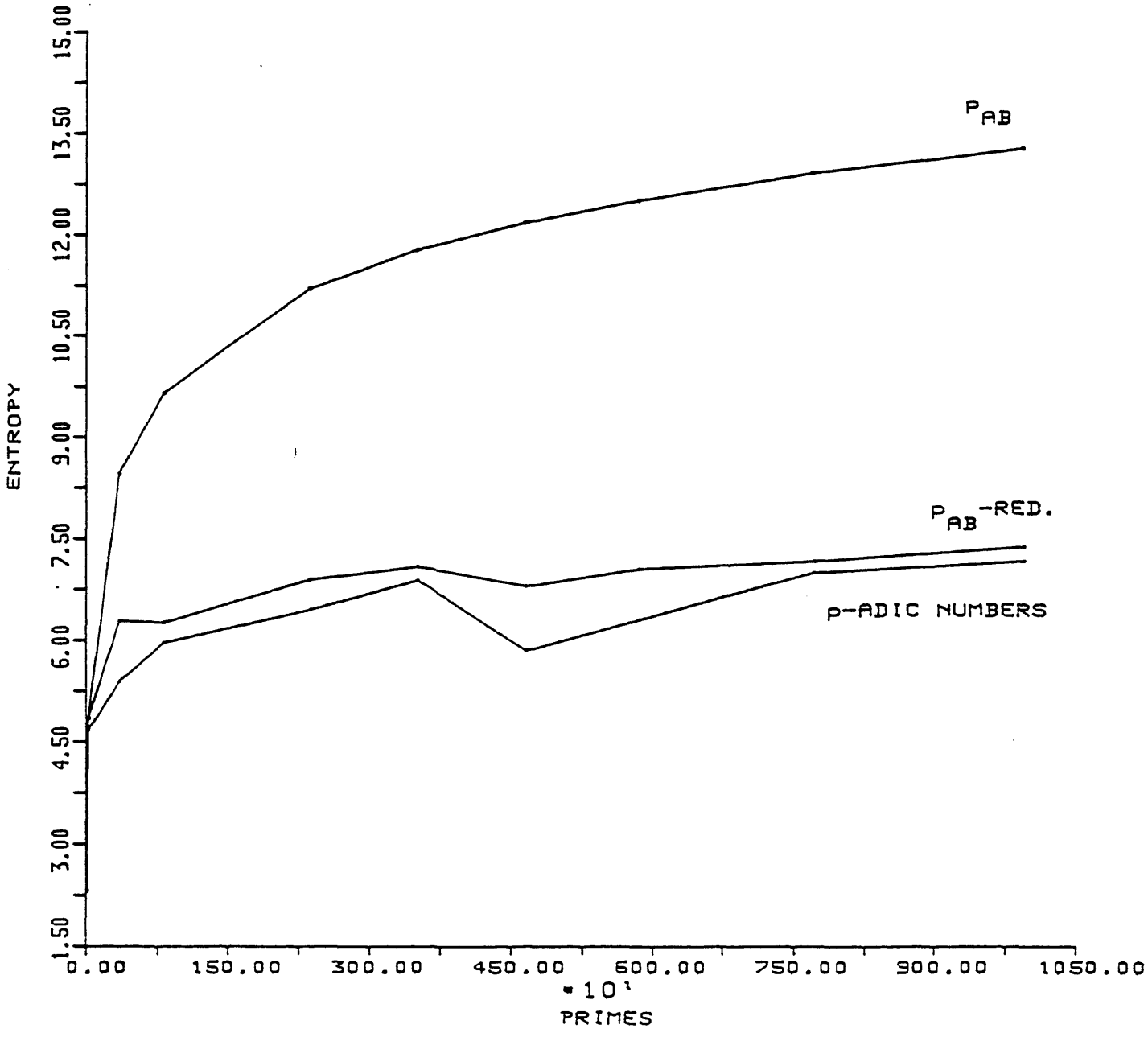


Fig.6.4 ENTROPY CURVES FOR RANDOM MESSAGES
WITH EQUIPROBABLE CHARACTER FREQS.
M = 100

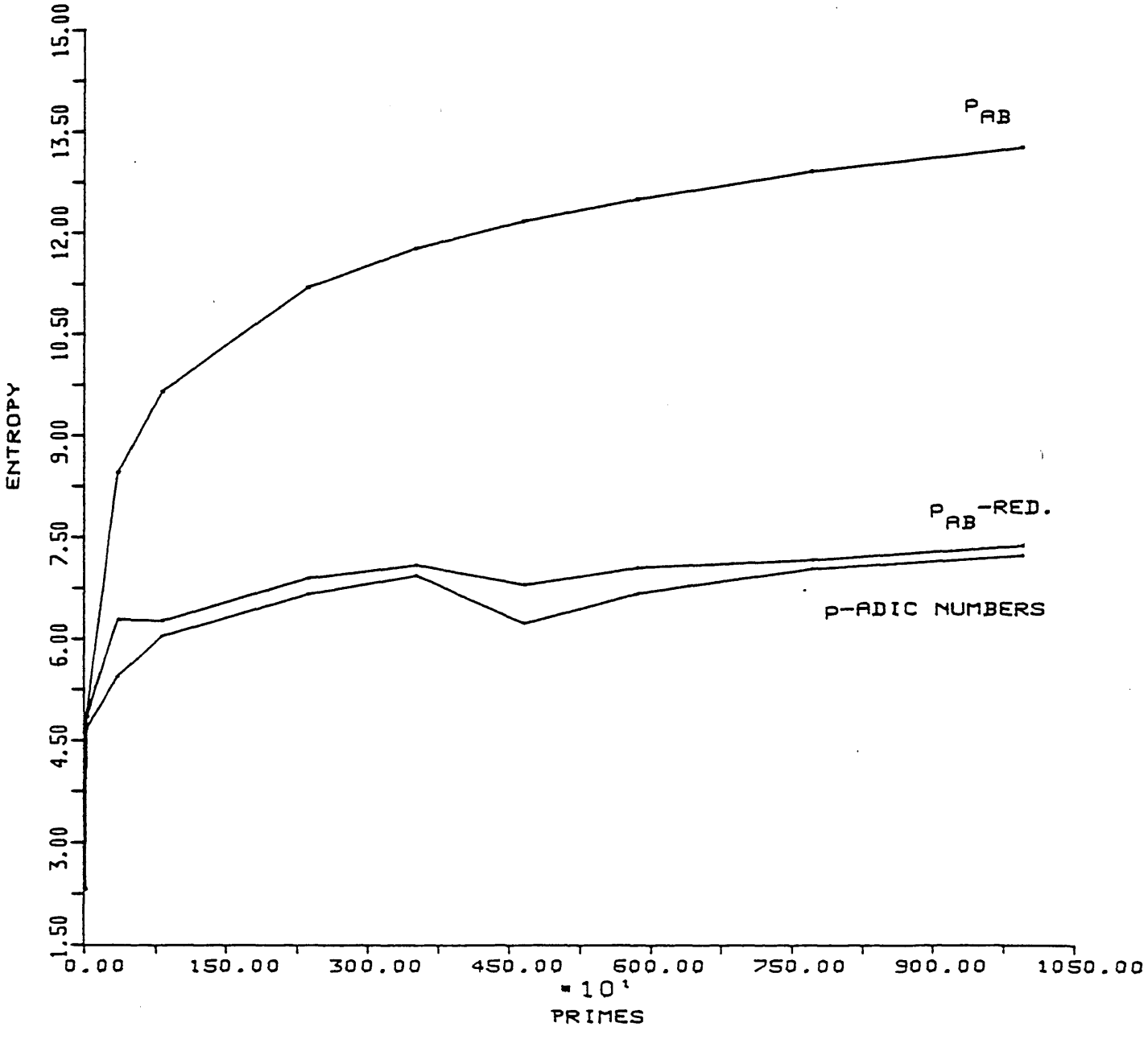


Fig.6.5 ENTROPY CURVES FOR RANDOM MESSAGES
WITH EQUIPROBABLE CHARACTER FREQS.
M = 1000

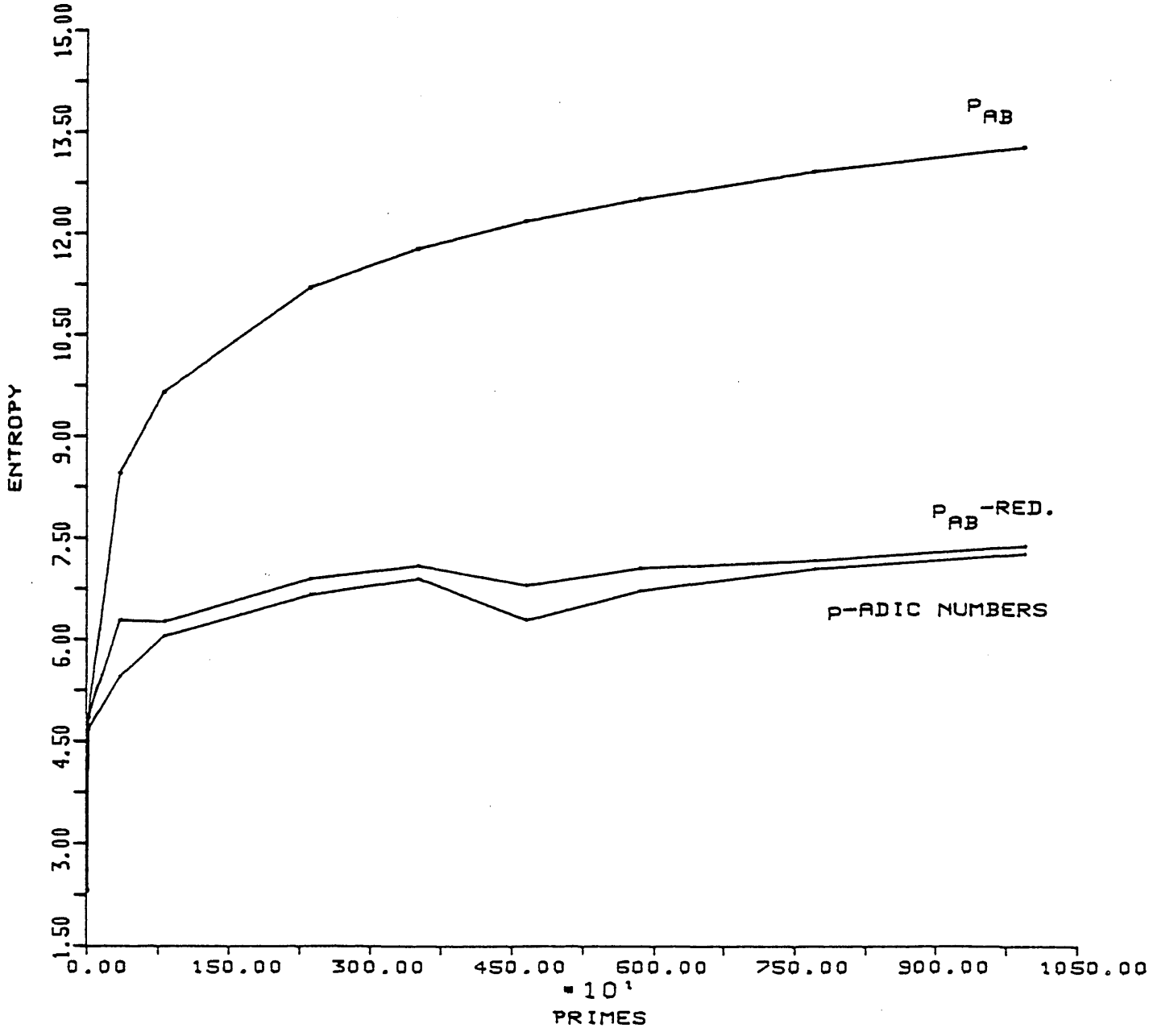


Fig.6.6 ENTROPY CURVES FOR RANDOM MESSAGES
WITH EQUIPROBABLE CHARACTER FREQS.
M = 10000

M = 100, 1000 and 10000 characters (based on a randomly generated alphabet with $\gamma = 20$), the entropy of the p-adic numbers occurring in the corresponding codes is respectively plotted.

The curve indicated by p_{AB} represents the upper bound for the corrresponding entropies of the p-adic numbers. This curve is equal to $\log_2 p_{AB}$ and is based on the fact that, for a prime p_{AB}, there are actually p_{AB} possible p-adic numbers and each of these numbers has an equally-likely probability of $\frac{1}{p_{AB}}$. Hence,

$$H(p_{AB}) = - \sum_{i=1}^{p_{AB}} p_i \log_2 p_i \qquad (6.2)$$

$$= \sum_{i=1}^{p_{AB}} \frac{1}{p_{AB}} \log_2 p_{AB}$$

$$= \log_2 p_{AB} \qquad (6.3)$$

The curve indicated by p_{AB}- RED. follows from the same reasoning, but additionally taking into consideration the fact that many of the p-adic numbers in the range 0 to $p_{AB} - 1$ do not appear in the code and hence this curve is the reduced form of the curve P_{AB}. Thus, it is shown that, even if a cryptanalyst could "filter" out the p-adic numbers which are not present in any length of message, all the other p-adic numbers occurring in the codes have an entropy approaching that of p_{AB} - RED. and, hence, the information content conveyed to the cryptanalyst through the ciphertext is decisively extremely uninformative.

In Figs. 6.7 - 6.12, the same analysis performed above is carried out but taking into consideration the relative frequencies of the English alphabet [7], [50] given in Table 6.1. For the prime p_{AB} = 2909, frequencies of occurrence of the p-adic numbers are shown in Figs. 6.7, 6.8 and 6.9 for simulated message lengths of M = 100, 1000 and 10000 respectively. Figs. 6.10, 6.11 and 6.12, on the other hand, show the entropy curves for the p-adic codes corresponding to various primes p_{AB} and compared with $\log_2 p_{AB}$ and $\log_2 (P_{AB}\text{-RED.})$. These 3 figures again correspond to M = 100, 1000 and 10000 characters, respectively.

The same conclusions reached previously hold in the case of the English alphabet, and it is apparent that a cryptanalyst's search to break the encryption algorithm, and based on statistical approaches is an infeasible task.

Furthermore, it is seen that no clear variations occur if longer message lengths are studied. Hence, even an attack directed at analysing the statistics of a "long" message will fail.

Next, we consider two specific messages in English and plot the frequencies of occurrence of the p-adic numbers appearing in the code corresponding to various values of p_{AB}. The two messages are "Making confusion worse confounded" and "It is a long road that has no turning". The plots corresponding to each of these messages are shown in Figs. 6.13 and 6.14 for p_{AB} = 2909, Figs. 6.15 and 6.16 for p_{AB} = 4001 and Figs. 6.17 and 6.18 for p_{AB} = 9967, respectively.

Table 6.1: Relative frequencies of English alphabetic characters and space

CHARACTER	RELATIVE FREQUENCY
space	19.248
A	6.595
B	1.205
C	2.246
D	3.434
E	10.257
F	1.799
G	1.627
H	4.921
I	5.625
J	0.124
K	0.624
L	3.251
M	1.943
N	5.450
O	6.062
P	1.558
Q	0.076
R	4.835
S	5.109
T	7.313
U	2.227
V	0.790
W	1.906
X	0.122
Y	1.594
Z	0.060

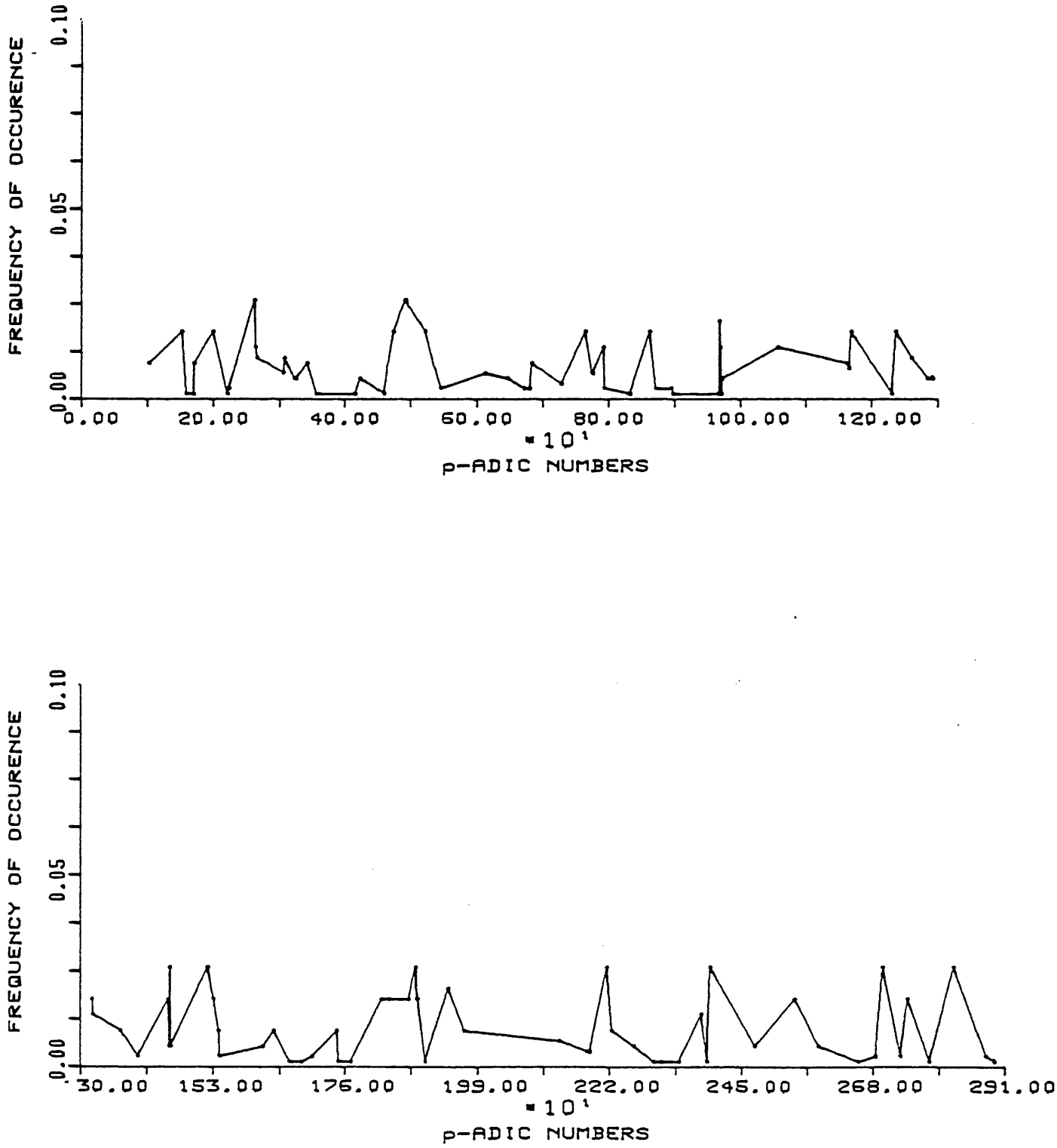
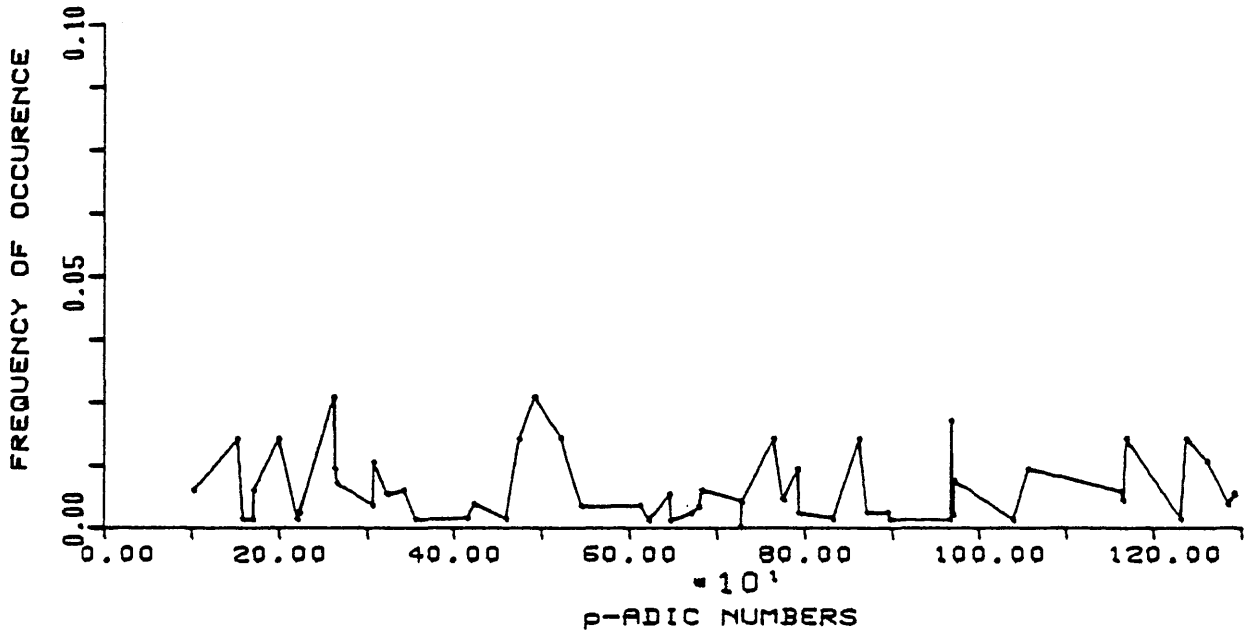


Fig.6.7 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH RELATIVE FREQS. OF ENGLISH CHARS. p_{AB} = 2909, M = 100



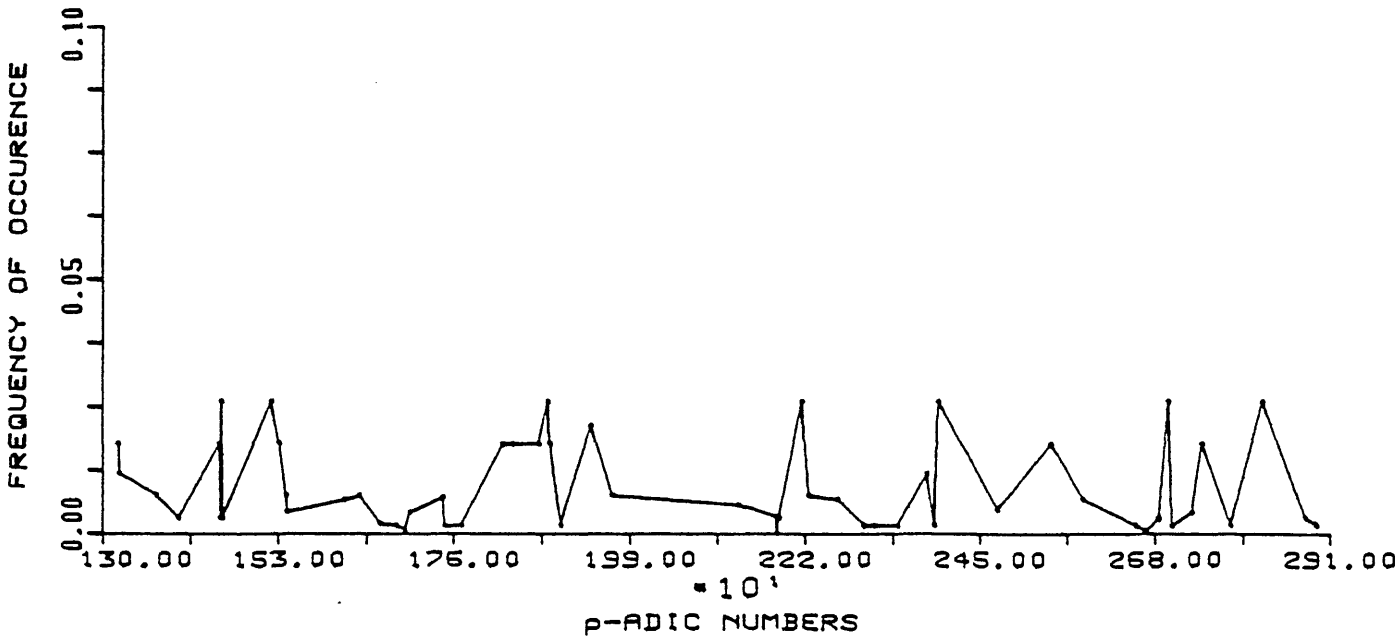
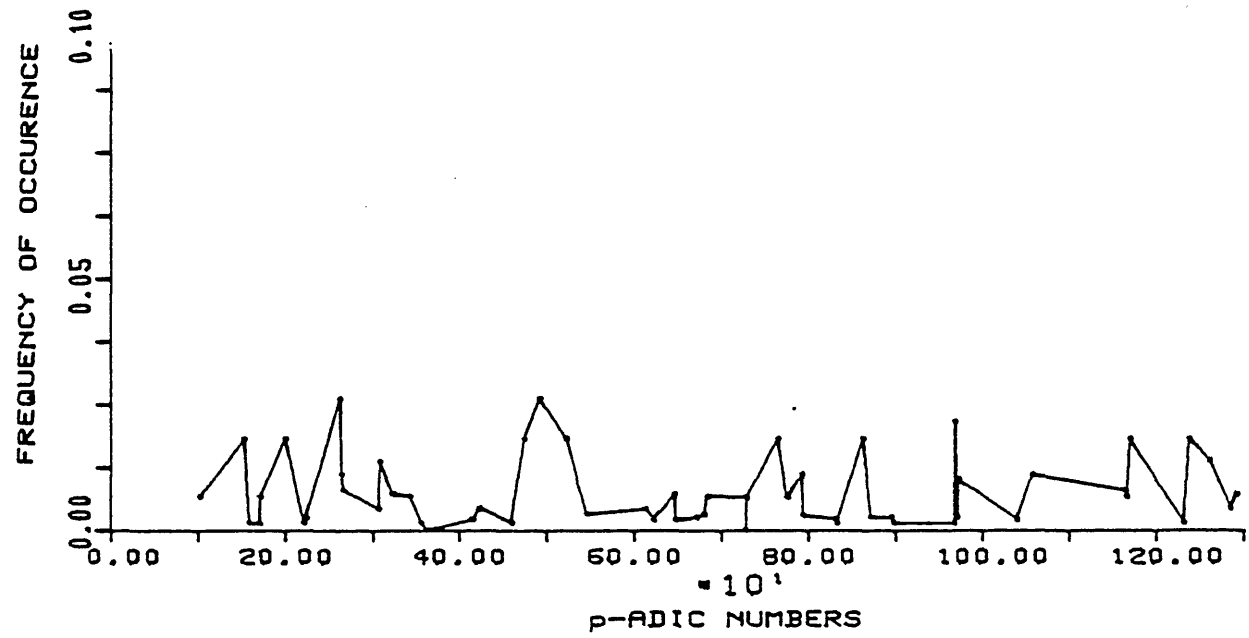


Fig.6.8 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH RELATIVE FREQS. OF ENGLISH CHARS.
P_{AB} = 2909, M = 1000



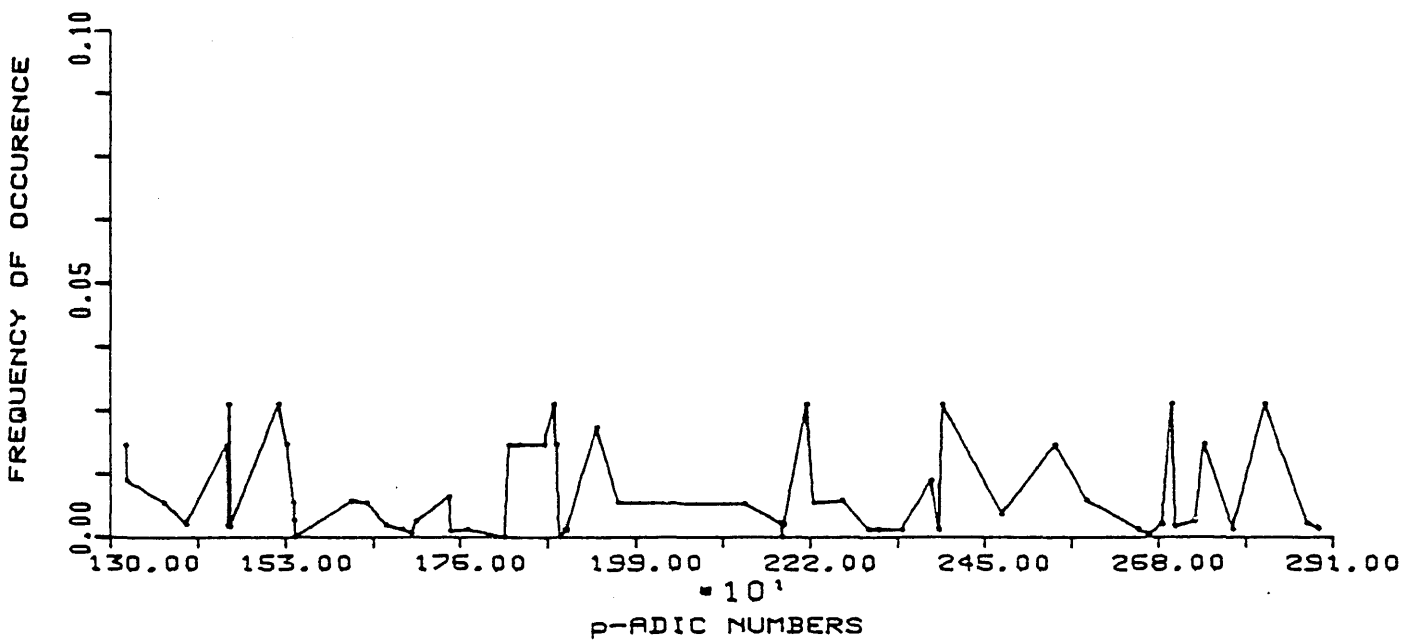


Fig.6.9 FREQUENCY OF OCCURENCE OF p-ADIC NUMBERS FOR RANDOM MESSAGE WITH RELATIVE FREQS. OF ENGLISH CHARS.
p_{AB} = 2909, M = 10000

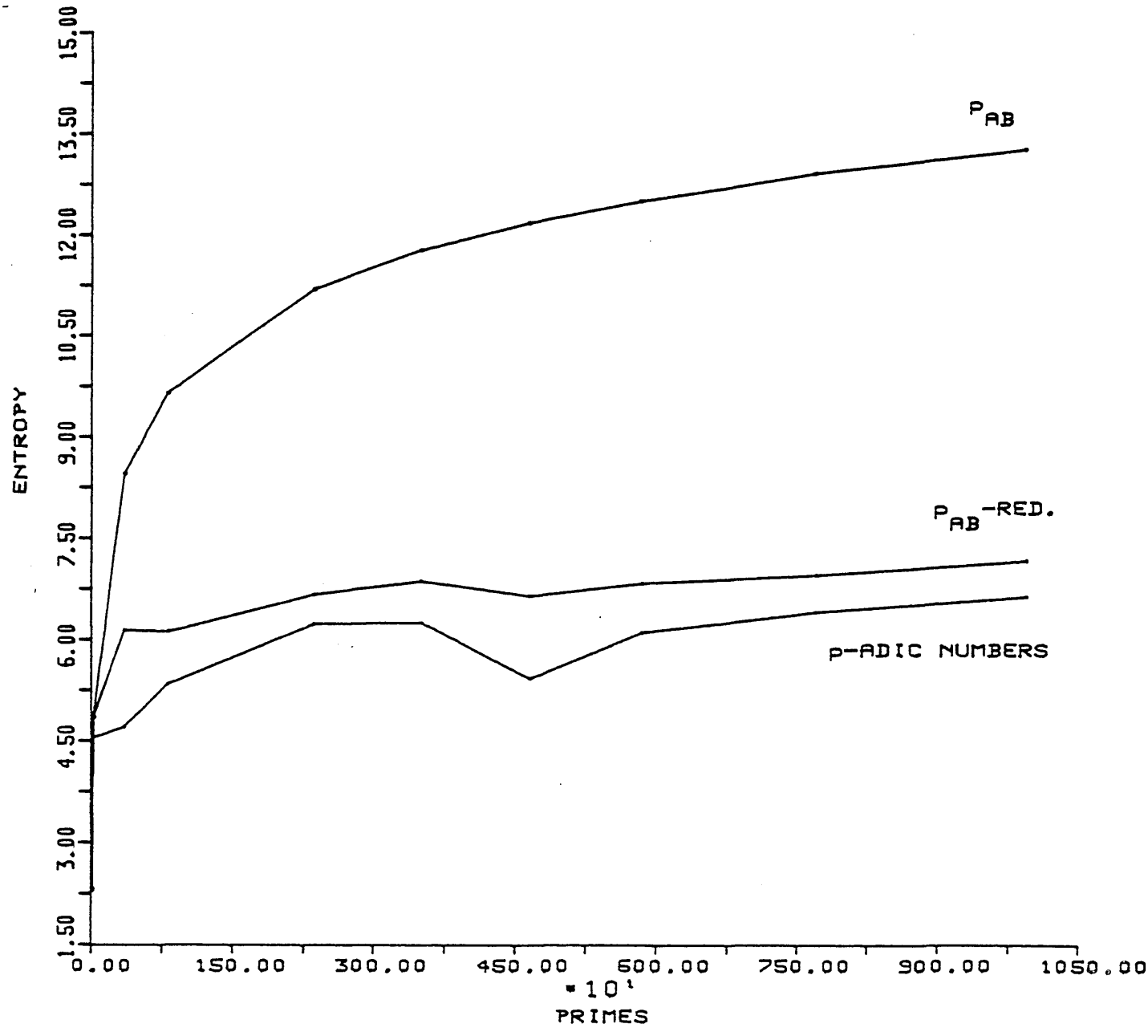


Fig.6.10 ENTROPY CURVES FOR RANDOM MESSAGES
WITH RELATIVE FREQS. OF ENGLISH CHARS.
M = 100

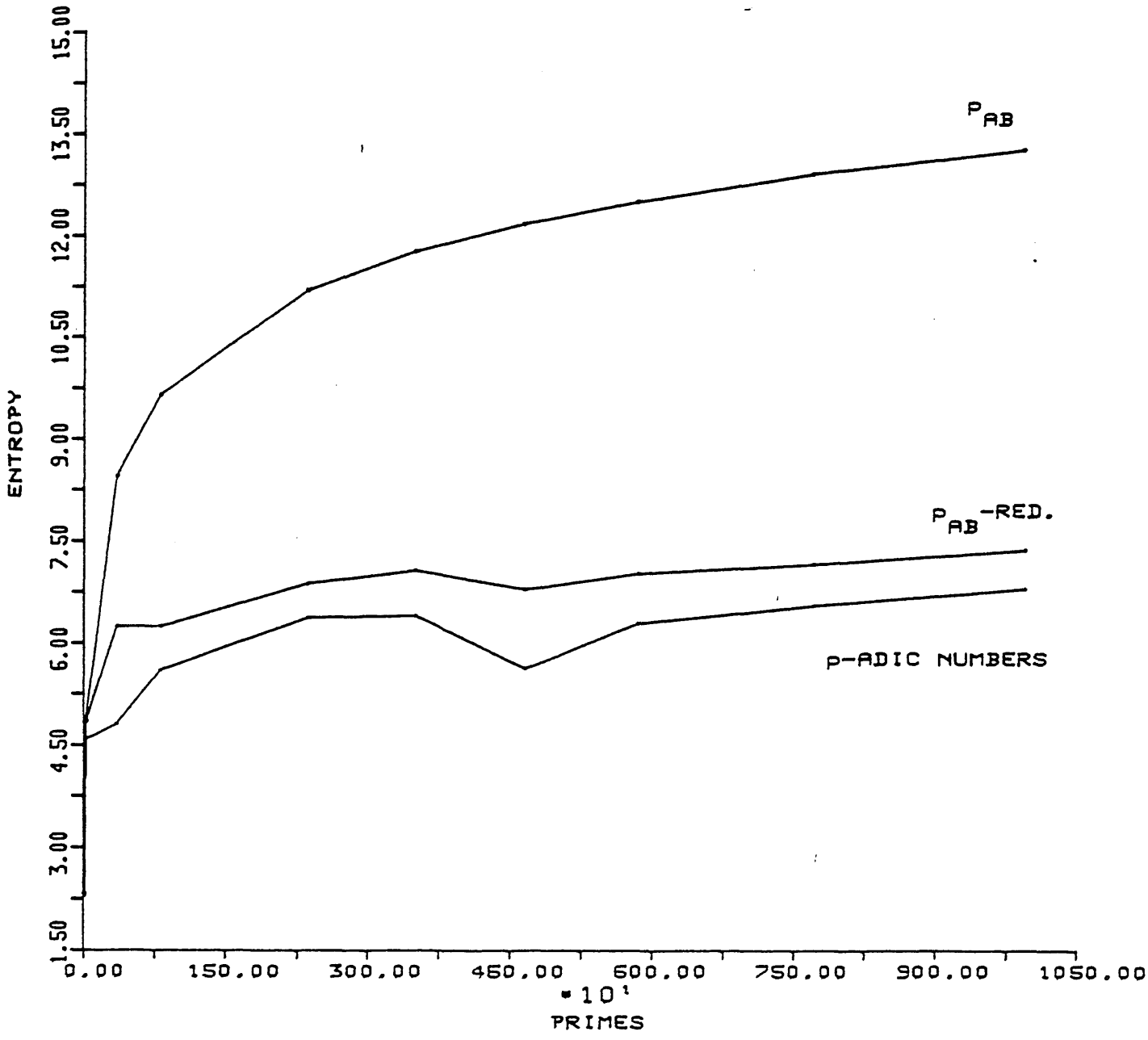


Fig.6.11 ENTROPY CURVES FOR RANDOM MESSAGES
WITH RELATIVE FREQS. OF ENGLISH CHARS.
M = 1000

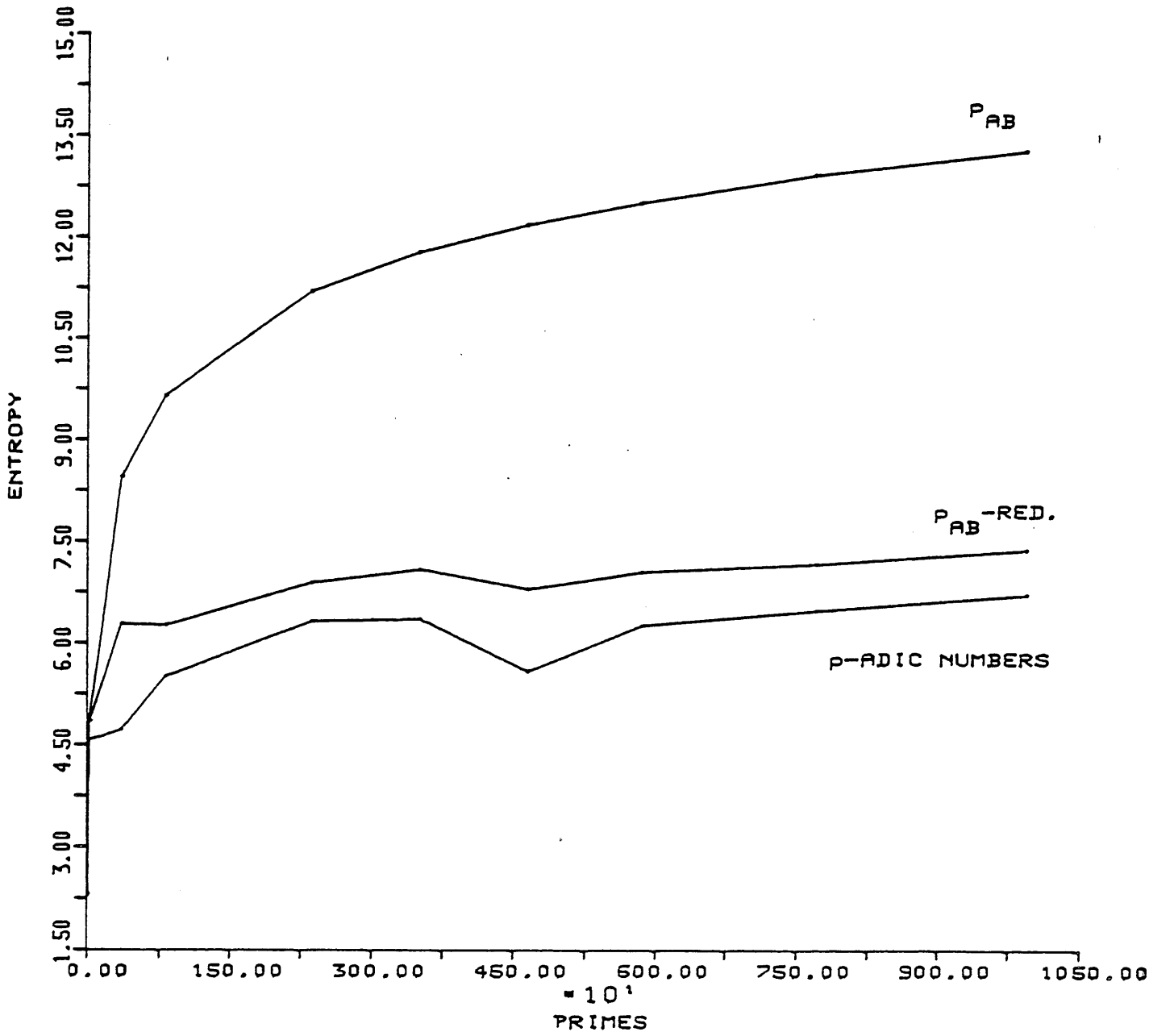


Fig.6.12 ENTROPY CURVES FOR RANDOM MESSAGES
WITH RELATIVE FREQS. OF ENGLISH CHARS.
M = 10000

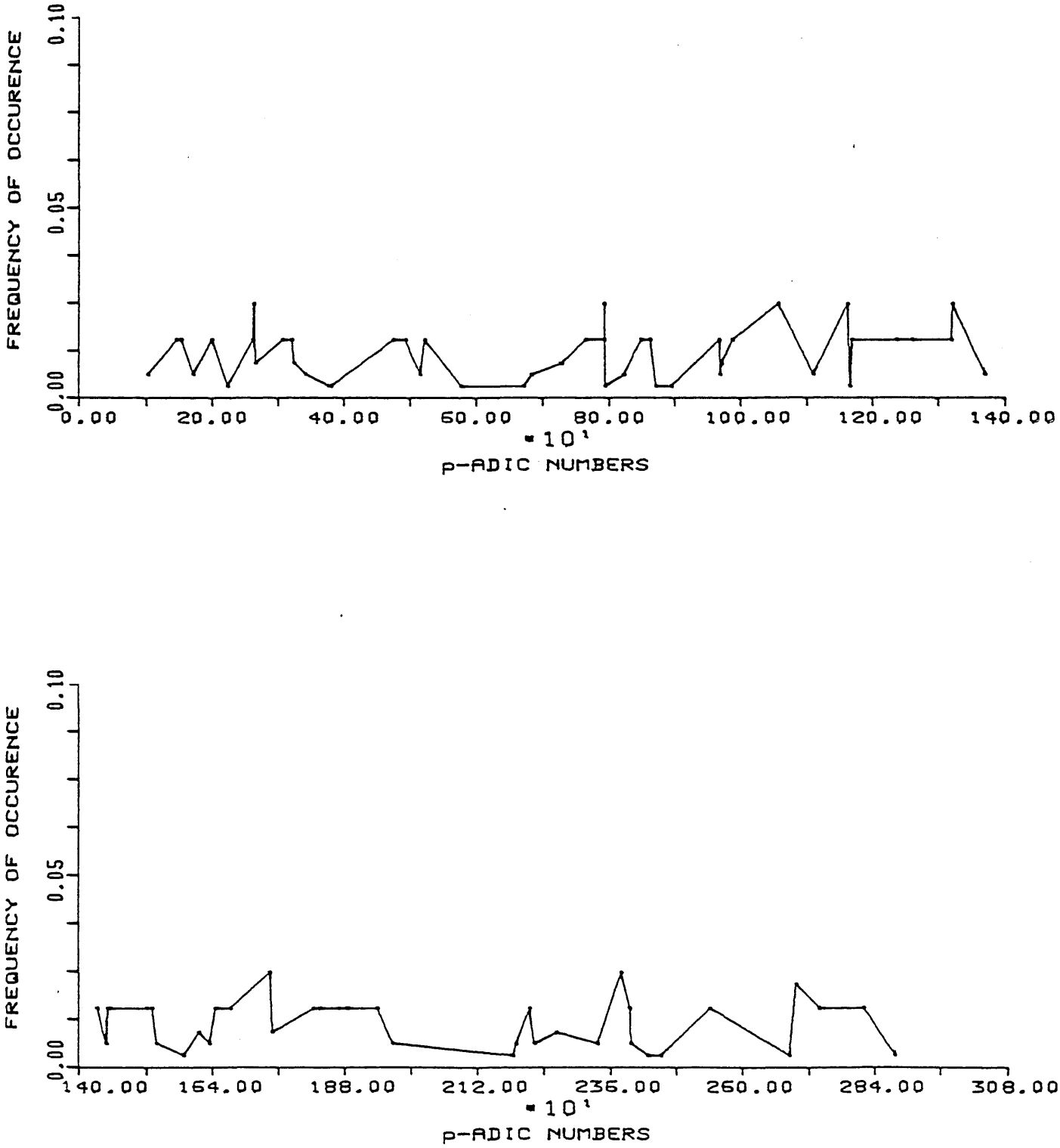
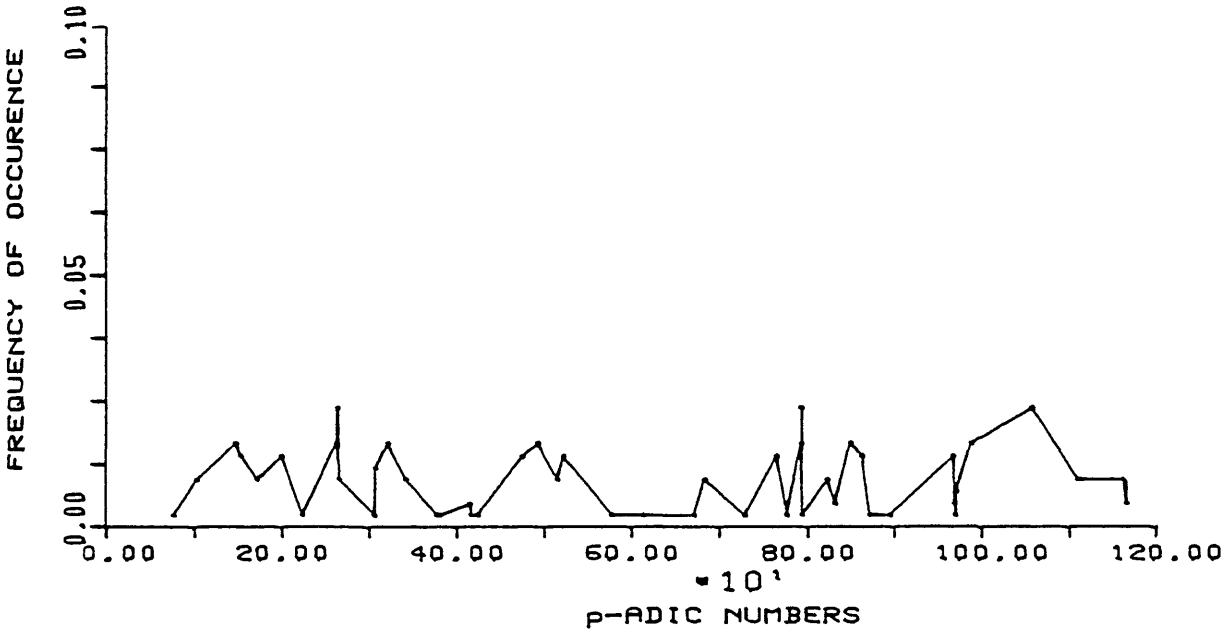


Fig.6.13 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE, MAKING CONFUSION WORSE CONFOUNDED.

P_{AB} = 2909



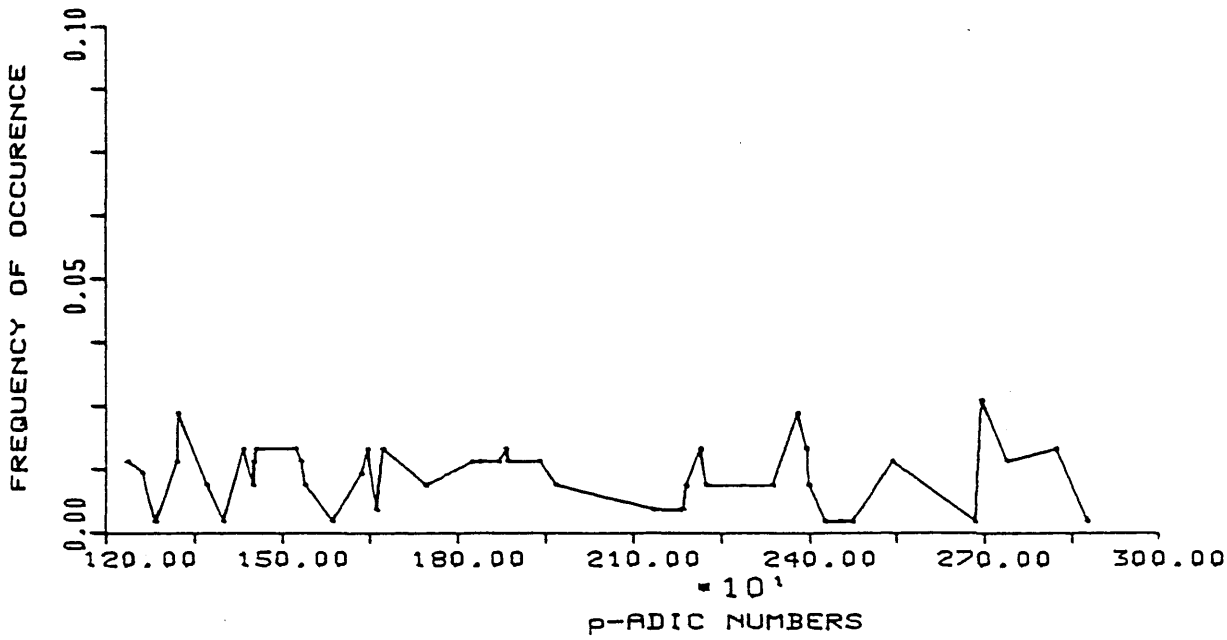
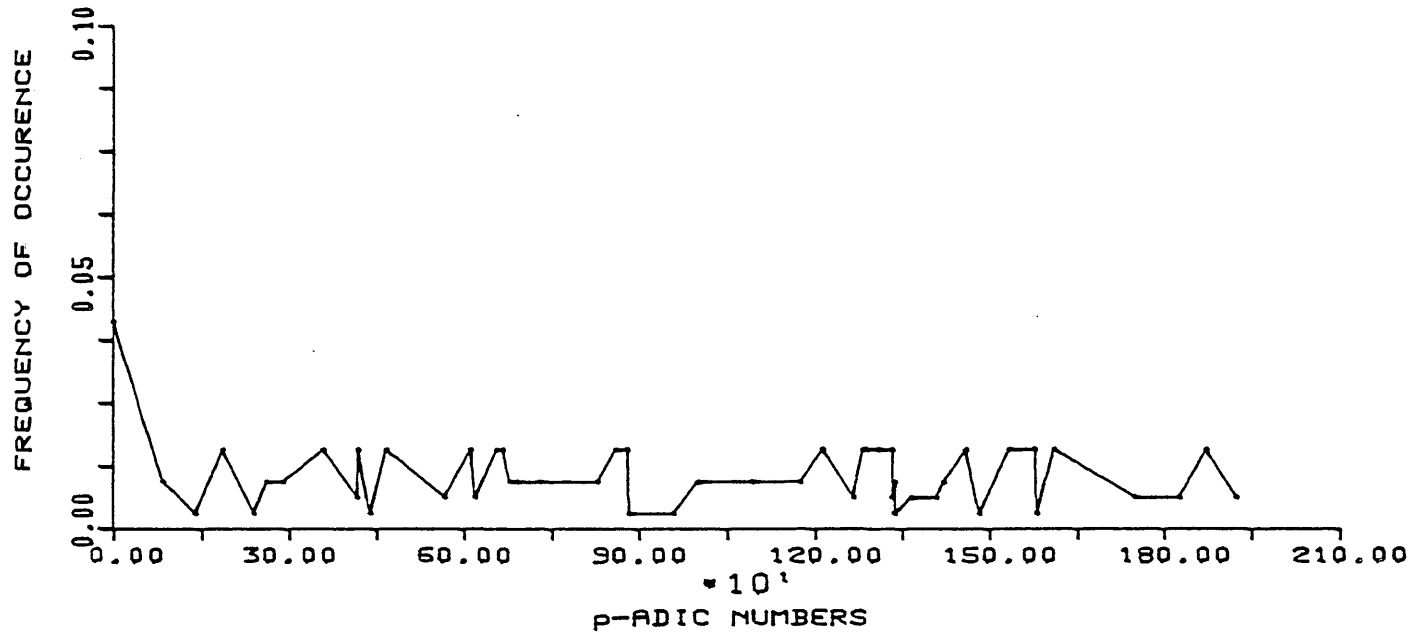


Fig.6.14 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE: IT IS A LONG ROAD THAT HAS NO TURNING

P_{AB} = 2909



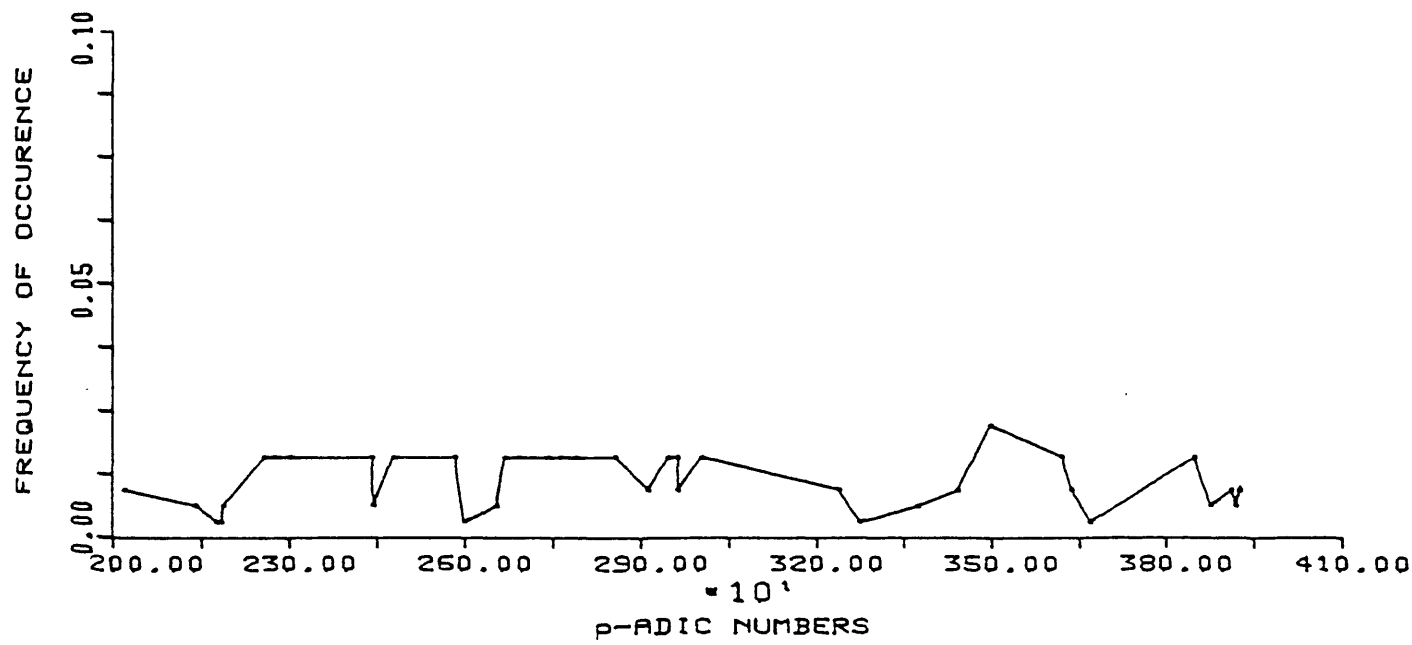
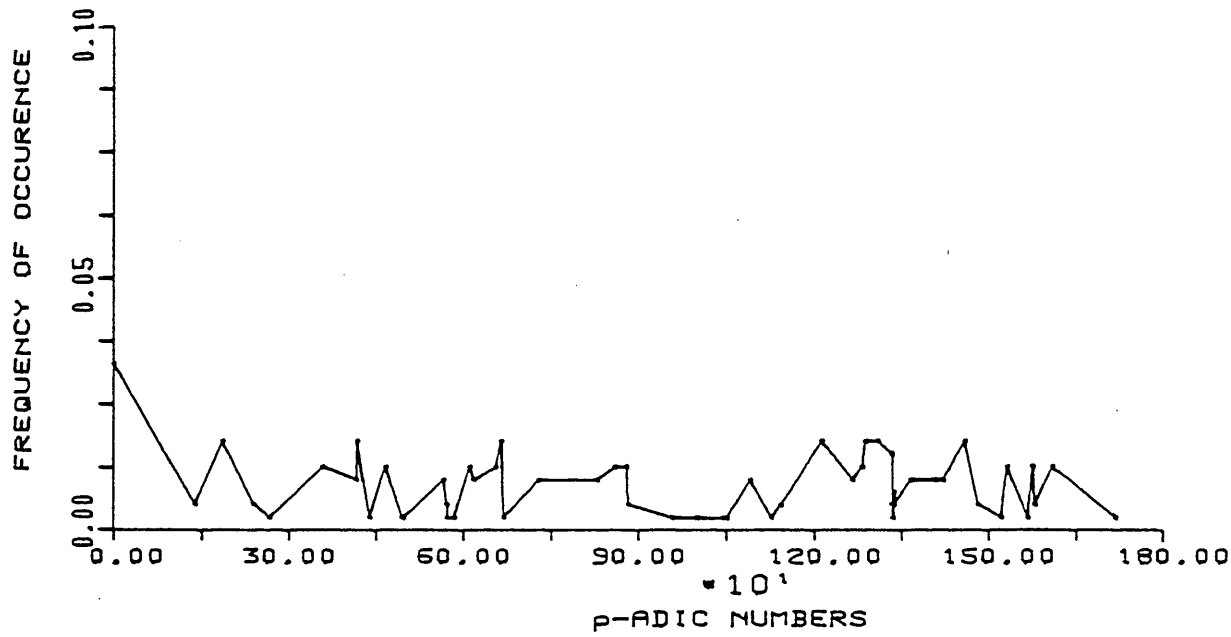


Fig.6.15 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE: MAKING CONFUSION WORSE CONFOUNDED.

$P_{AB} = 4001$



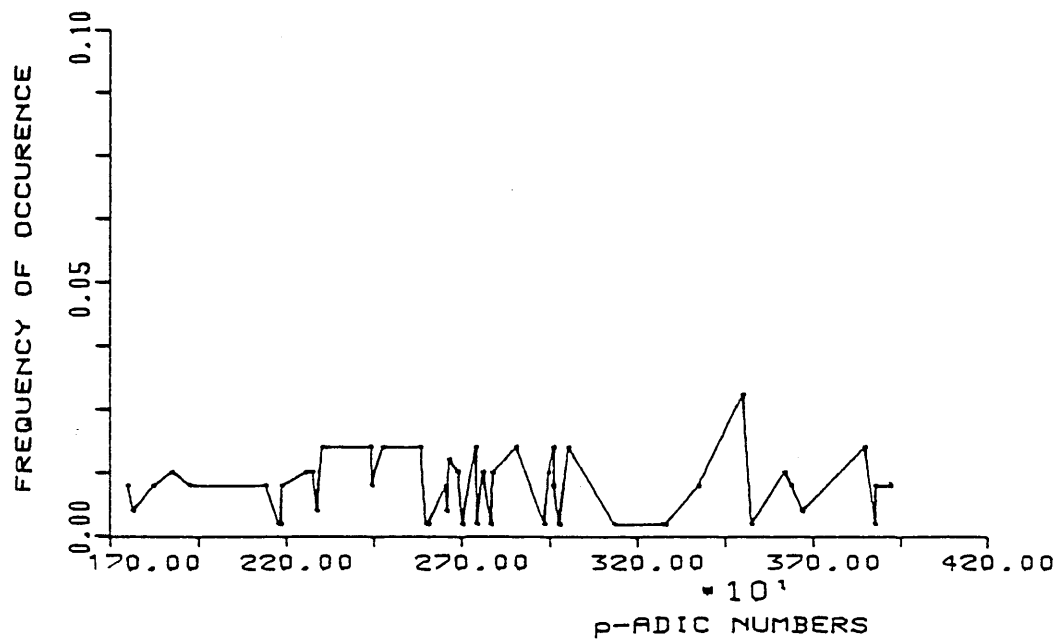


Fig.6.16 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE, IT IS A LONG ROAD THAT HAS NO TURNING

P_{AB} = 4001

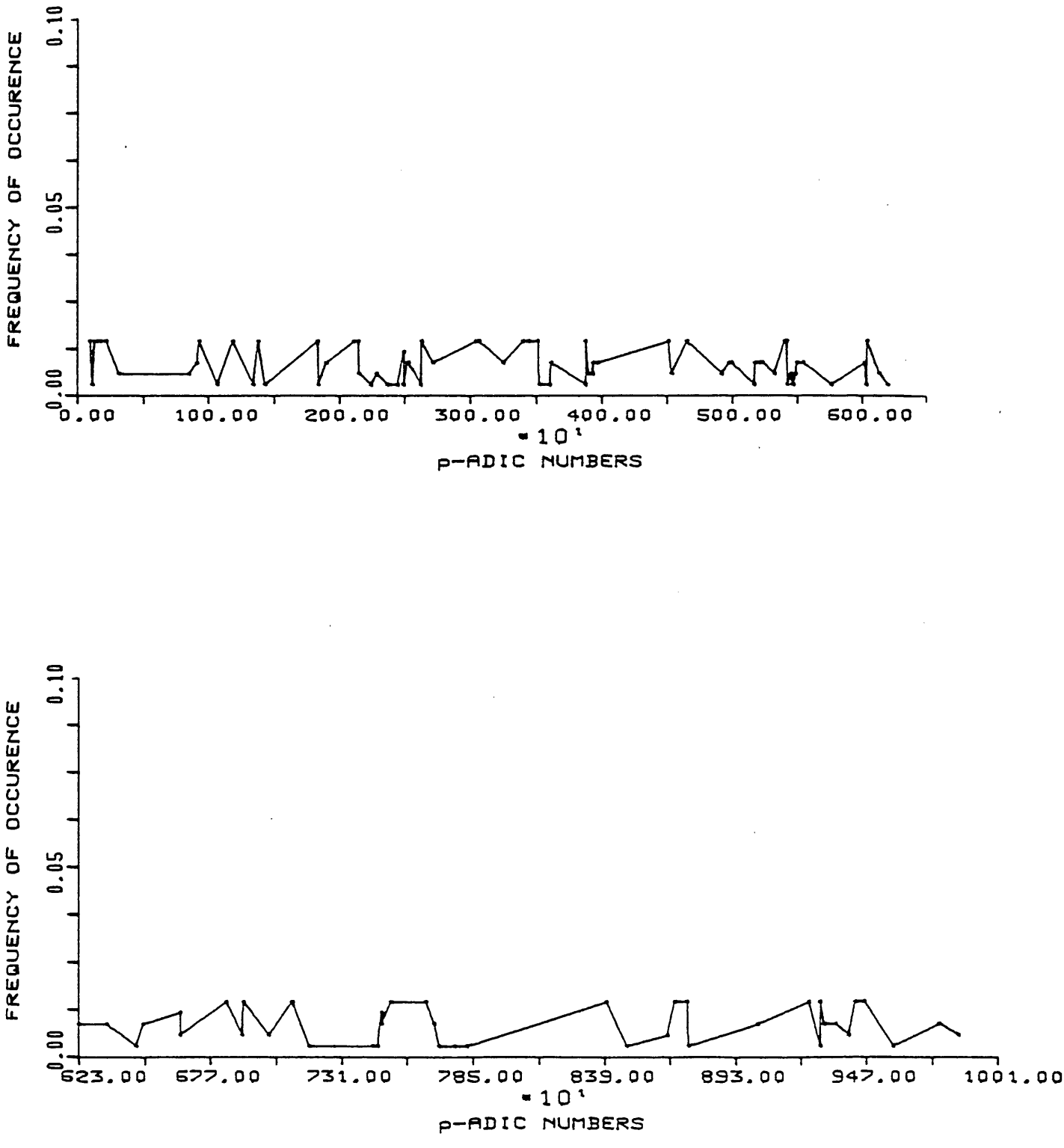


Fig.6.17 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE: MAKING CONFUSION WORSE CONFOUNDED.

P_{AB} = 9967

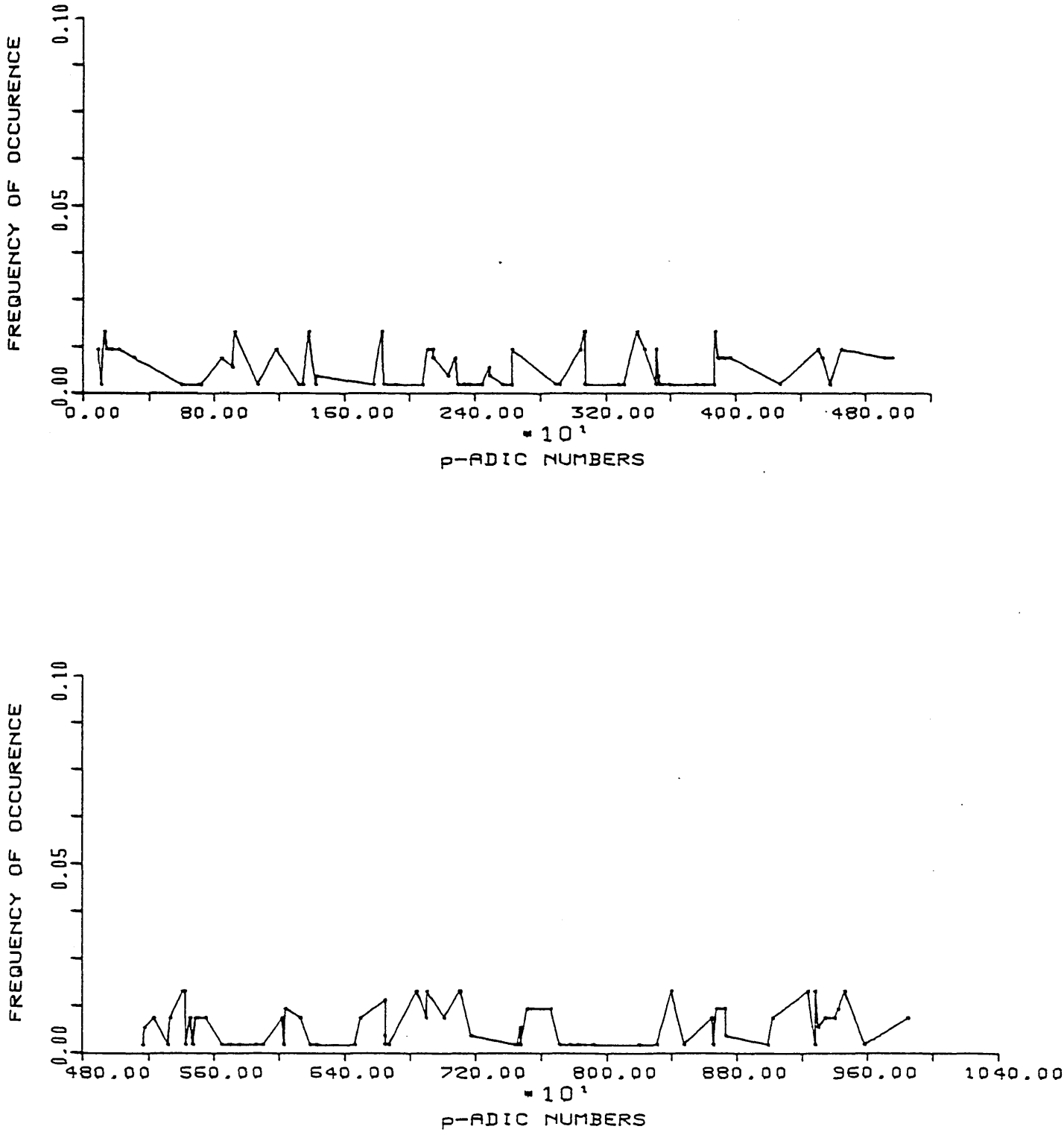


Fig.6.18 FREQ. OF OCCURENCE OF p-ADIC NUMBERS FOR THE MESSAGE, IT IS A LONG ROAD THAT HAS NO TURNING

P_{AB} = 9967

It is shown that in each of these plots, the p-adic numbers which are present in the range 0 to $p_{AB} - 1$ have a random distribution in all cases. Such a distribution does not lend an intruder with any particular information and a systematic search over various p-adic fields $Q_{p_{AB}}$ is rather a tedious task.

Moreover, it is important to take into consideration the fact that the randomly generated alphabet in all the previous instances only considered an alphabet consisting of 26 letters and a space. If the size of the alphabet is increased, so will increase the number of p-adic numbers and the additionally created elements of the code will contribute to the increase of the degree of randomness in the distribution.

Before moving to the evaluation of the g-adic scheme, one particular point is analysed in detail.

In the proposed p-adic code structure it was suggested that the first two elements in the code refer to the periodicity and to its start, respectively.

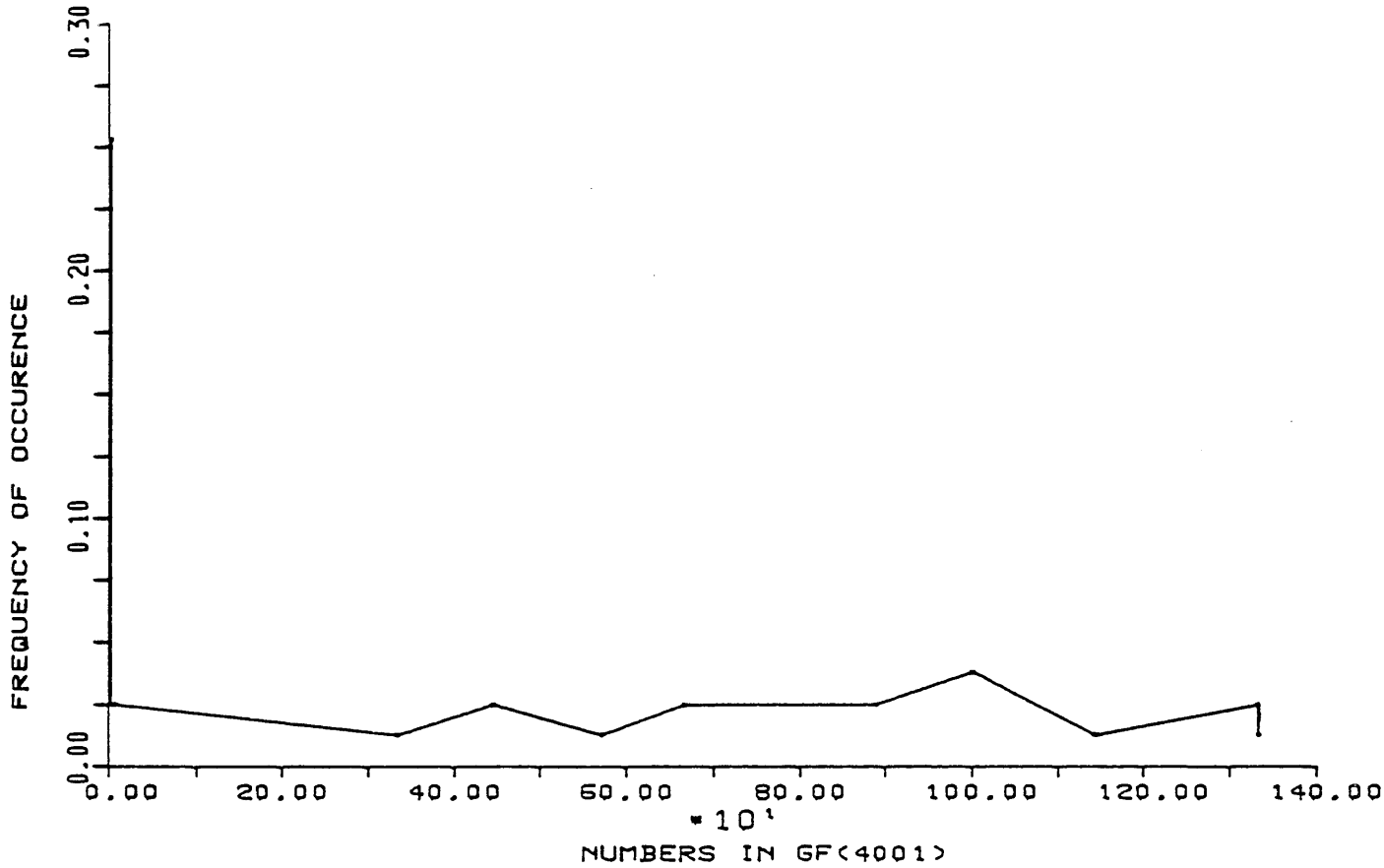
When using large values of p_{AB}, the corresponding values of λ and λ_1 will generally have a much smaller magnitude than that of the elements in the p-adic code. Furthermore, values of λ and λ_1 are repetitive in the sense that, given p_{AB}, many denominators in the alphabet assume the same values of λ or λ_1 (refer to Tables 2.1(A) and 2.1(B)) and hence these values have a higher frequency of occurrence than that of the p-adic numbers themselves.

This is illustrated in the following ciphertext generated from the message: "Its all Greek to me" (first used by Rivest et. al. [46] and which was shown to be breakable under the RSA system in section 4.4.6). Here, a randomly generated alphabet and the prime $p_{AB} = 4001$ are used. The ciphertext is then composed of blocks of 4-digit numbers and the markers indicate the value of λ at the start of each sequence:

0002000213342667133300060002044508891778355631112222044400010002250125000003000234302857171434290002000233350666333400020002333506663334000200022668133326670002000203341667033300010002100110000001000210011000000100023502350000060002044508891778355631112222044400030002114422860571114300010002160216000001000210011000

To emphasize this fact, the plot of Fig. 6.19 shows, although not very clearly, that a peak occurs, in the frequency distribution curve, at the values of 0001 and 0002, due to the repetitive occurrences of λ and λ_1. The same fact is clarified through Fig. 6.20 where the frequency of distribution of the p-adic numbers is plotted against their logarithms. The above-mentioned peak is then clearly seen.

To avoid any risk that such sharp boundaries which occur between successive sequences may give any indications, however extremely small, to the value of the prime p_{AB}, it is suggested that values of λ and λ_1 be embedded in the code itself. This is achieved as follows.



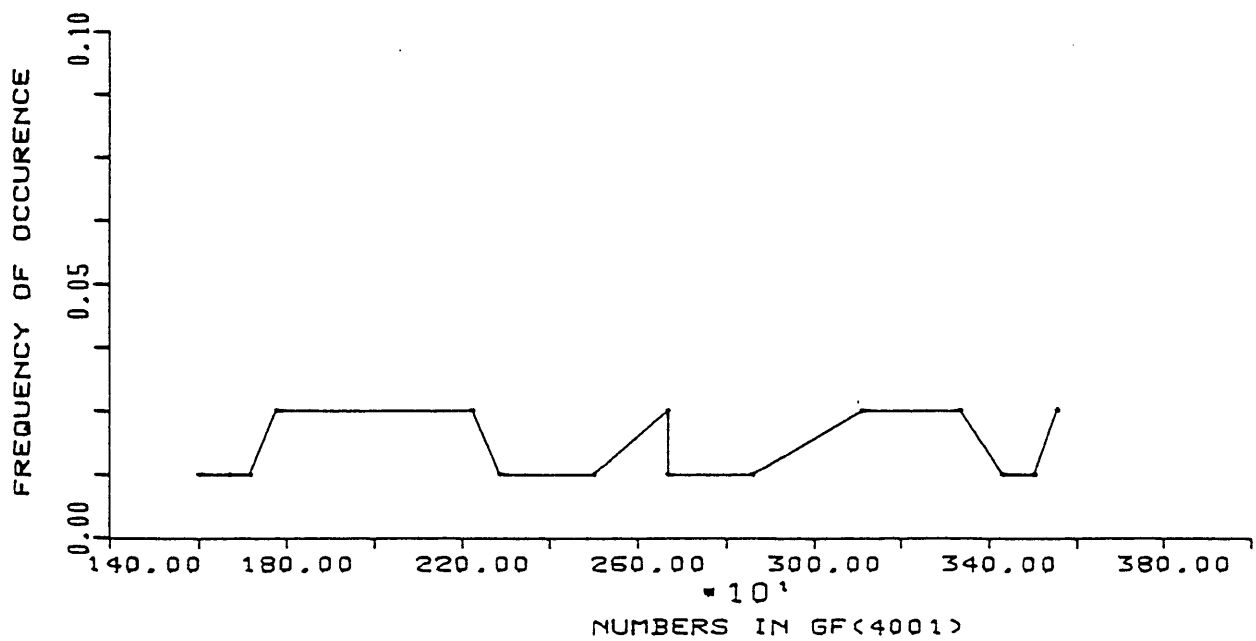


Fig.6.19 FREQ. OF OCCURENCE OF p-ADIC NOs. AND PERIOD PARAMETERS FOR THE MESSAGE; ITS ALL GREEK TO ME

P_{AB} = 4001

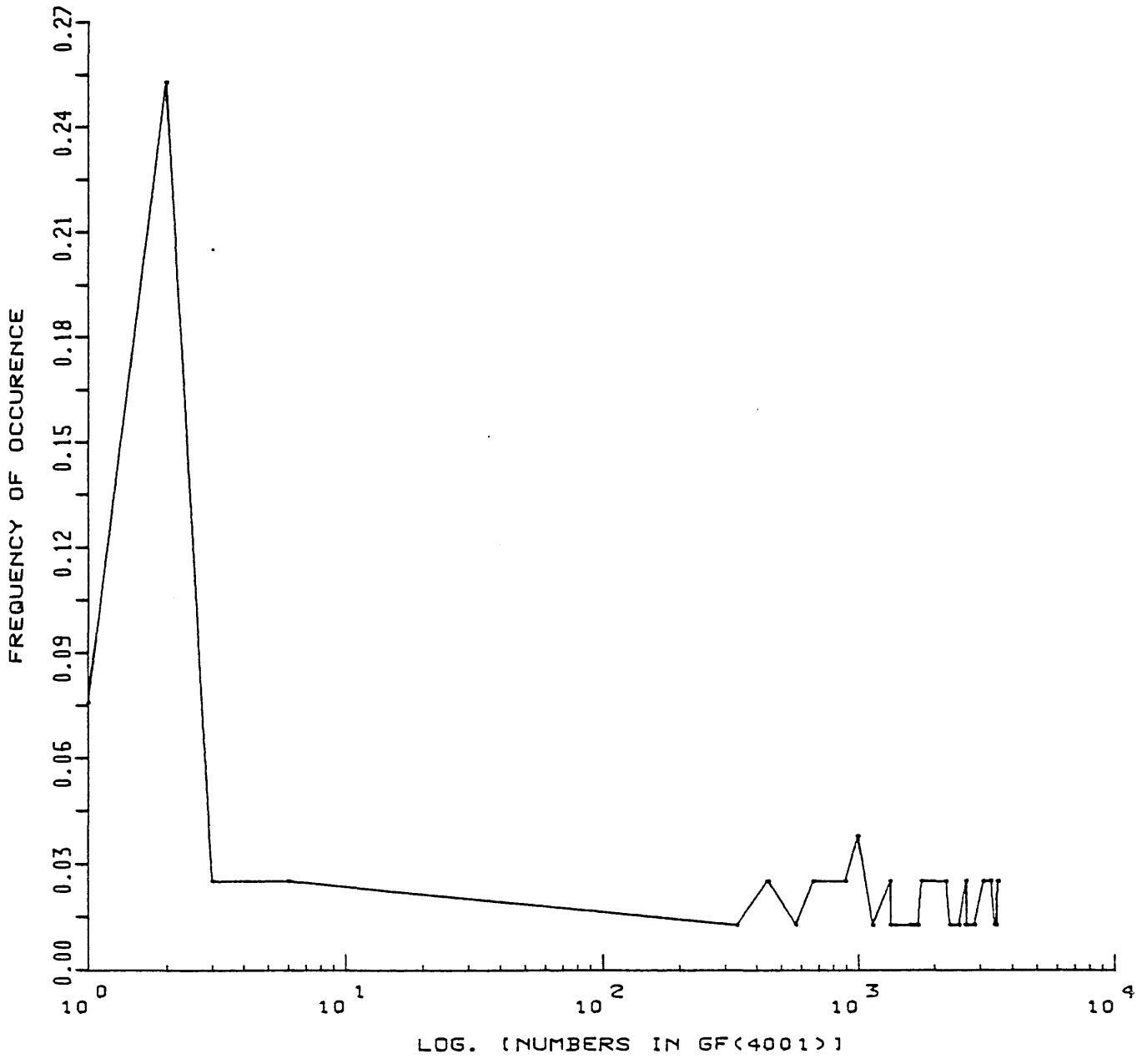


Fig.6.20 FREQ. OF OCCURENCE OF p-ADIC NOs. AND PERIOD PAR'S (LOG.SCALE) FOR THE MESSAGE, ITS ALL GREEK TO ME. - p_{AB} = 4001

In the encryption algorithm, instead of sending λ and λ_1 explicitely in the code, it is proposed that the sender keeps a running counter, N, of each element in the plaintext. This element is increased by 1 for λ, then for λ_1, and then consecutively for their next occurrences. Hence, the sender A transmits the values

$$\lambda' \equiv p_{AB} \cdot \lambda \cdot N \pmod{p} \tag{6.4}$$

and

$$\lambda_1' \equiv p_{AB} \cdot \lambda_1 \cdot (N+1) \pmod{p} \tag{6.5}$$

instead of actually transmitting λ and λ_1. The next value of N will then be N+2 for the following λ', and so forth.

The recipient, B, on the other hand, has already computed p_{AB} from the header and, starting with a value of N = 1, B can keep track of the same running counter N. Hence, B can compute λ and λ_1 from λ' and λ_1' respectively, such that:

$$\lambda \equiv \lambda' \cdot [p_{AB} \cdot N]^{-1} \pmod{p} \tag{6.6}$$

and

$$\lambda_1 \equiv \lambda_1' \cdot [p_{AB} \cdot (N+1)]^{-1} \pmod{p} \tag{6.7}$$

The inverse values modulo p in congruences (6.6) and (6.7) are guaranteed to exist since p is a prime (Euler's theorem).

Consequently, B will be the sole detector of the periodicity parameters through the whole of the transmitted message and thus can exactly and secretly determine the previously distinctive boundaries.

In conclusion, this scheme achieves two goals:

1) As mentioned above, the sharp edges which indicate the start and end of a sequence of ciphertext are "smoothed" out. A cryptanalyst cannot derive any information regarding the structure of the transmitted ciphertext and, hence, the very minimal amount of information which was previously present in the code has now completely disappeared.

2) Since the values of λ' and λ_1' are computed modulo p (and not p_{AB}), then the numbers appearing in the ciphertext will be randomly distributed over the range 0 to p-1, thus increasing the search space which will have to be considered by a cryptanalyst to break the code.

The ideas discussed above are best described through the same example considered earlier. In this case, p = 7489 and p_{AB} = 4001. The code is then as follows:

0513102613342667133364822052044508891778355631112222044450274943250125 00
3507596934302857171434296482699533350666333400190532333506663334291034 23
2668133326673936444903341667033324815475100110002994650110011000350719 03
3502350053832929044508891778355631112222044423513955114422860571114369 11
4981160216007424038310011000

It is noticed that, although we have retained the same markers positions for ease of locating the start of each sequence, these markers now point out at the value of λ' (the next value being that of λ_1').

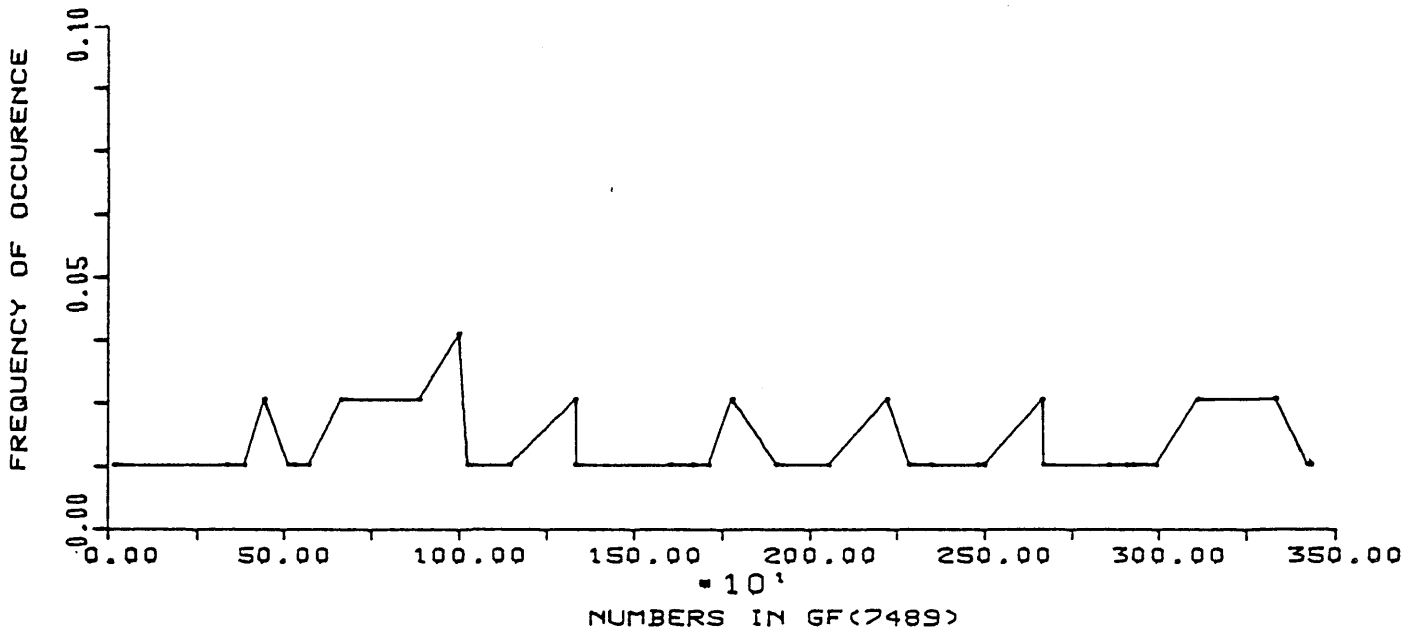
The frequencies of occurrence of the numbers present in the ciphertext, and which are no longer in $Q_{p_{AB}}$, but in the much larger field of GF(P), are plotted in Fig. 6.21 for p = 7489 and p_{AB} = 4001. It is shown that the previous peak no longer exists in the new ciphertext and to emphasize this observation, the same frequencies of occurrence are again plotted in Fig. 6.22, but against the logarithms of the numbers occurring in the code and which belong to GF(P).

6.2 Attempts at Breaking the g-adic-Based System - Diffusion and Confusion

Having subjected the p-adic-based cryptosystem to the attacks of the previous section, the g-adic scheme is now analysed for security.

To break the scheme based on g-adic sequences, two factorization problems of the headers h_1 and h_2 have to be solved to recover the values of p_{AB} and n. Both values are necessary to decipher the transmitted ciphertext.

Without the knowledge of p_{AB} and n, if a cryptanalyst reverts to a statistical attack on the ciphertext, then the analysis of the g-adic system is much more complicated than that of the p-adic system. This is due to the fact that the search space with the g-adic scheme spans over Q_g where $g = p_{AB}^n$ and $n \in Z^+$. In other words, the search



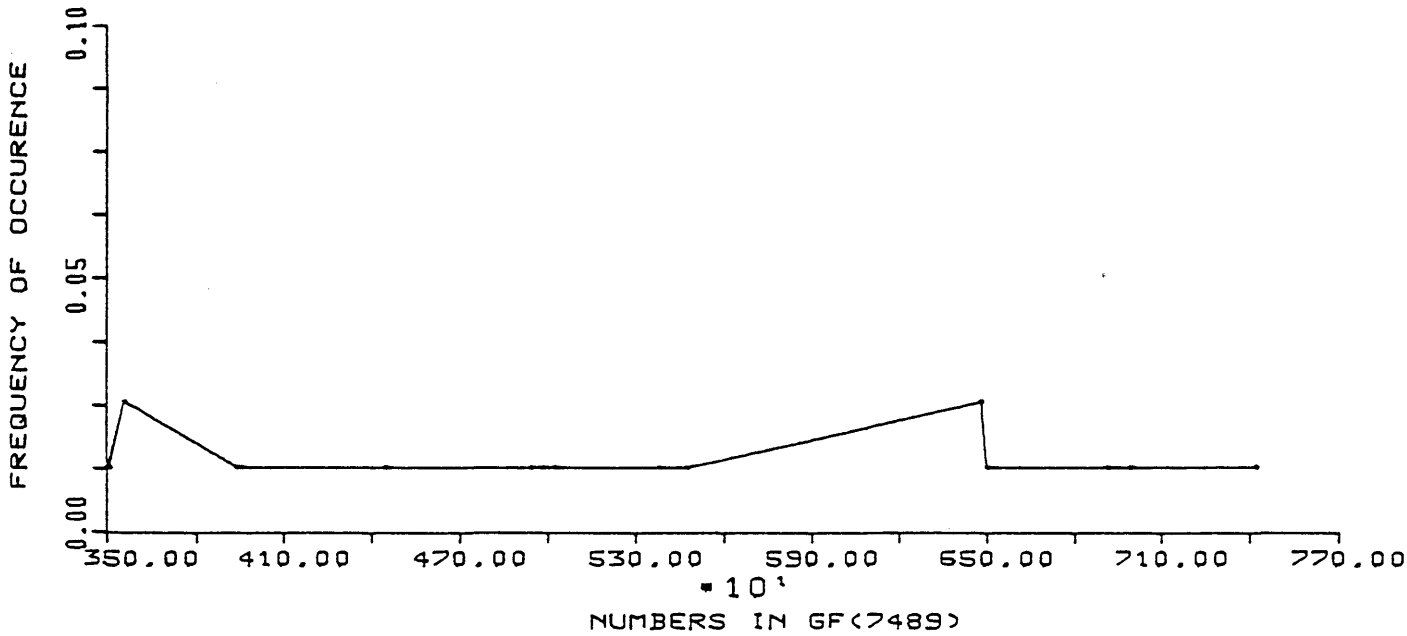


Fig.6.21 FREQ. OF OCCURENCE OF p-ADIC NOs. AND MODIFIED PERIOD PAR'S FOR THE MESSAGE, ITS ALL GREEK TO ME. - p = 7489, p_{AB} = 4001

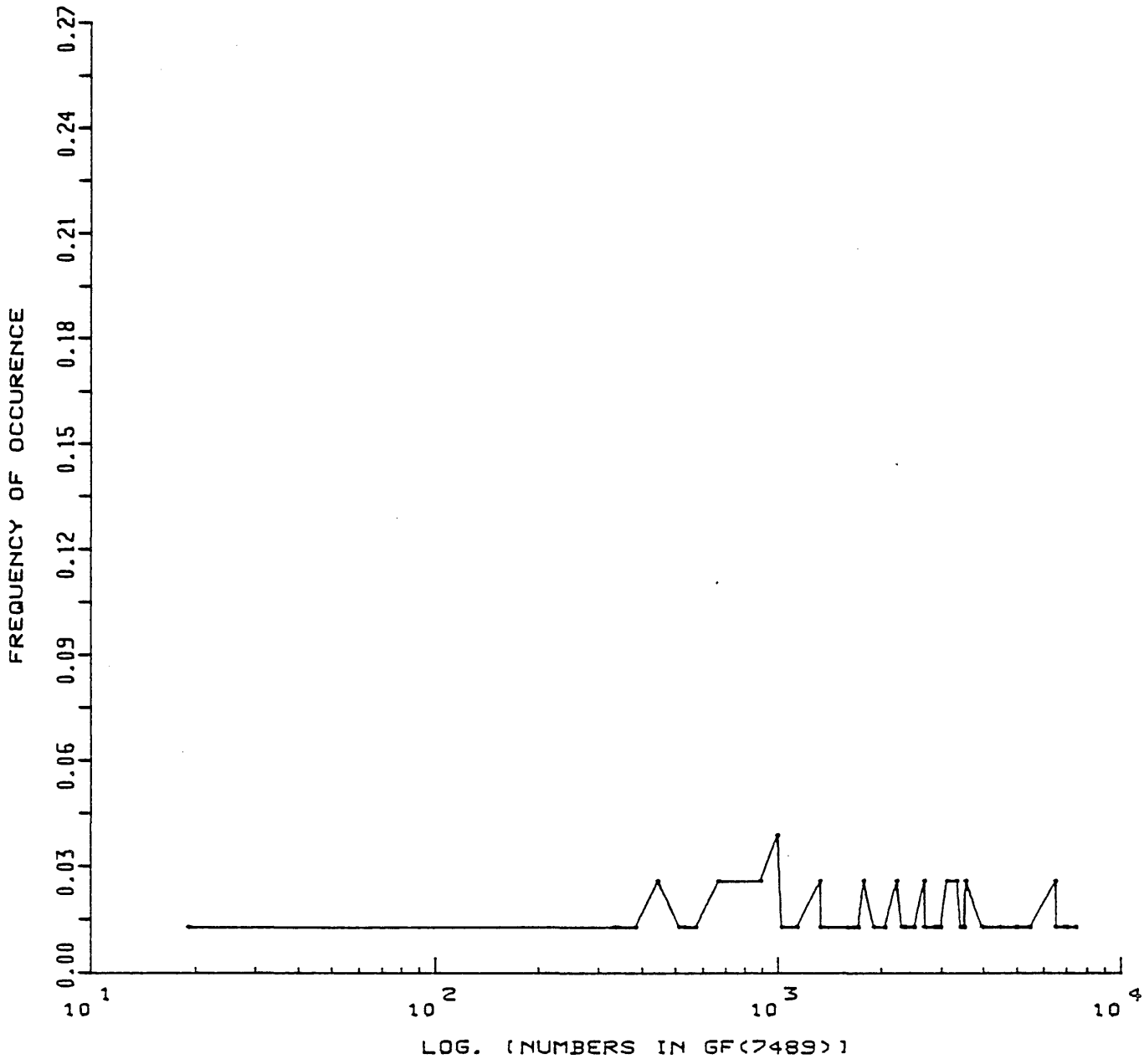


Fig.6.22 FREQ. OF OCCURENCE OF p-ADIC NOs. AND MODIFIED PERIOD PAR'S (LOG. SCALE) FOR THE MESSAGE, ITS ALL GREEK TO ME.
$p = 7489$, $p_{AB} = 4001$

space increases exponentially with increasing values of n. Such a search is computationally infeasible since the only bound on n is that $n \leq p-1$.

Furthermore, the different combinations of p_{AB} and n are unlimited in the sense of a computer search. Consequently, these aspects lend the proposed system to the class NP.

Shannon [49] introduced the notion of diffusion. In this method, the statistical structure of a message which leads to its redundancy is dissipated into long range statistics. A cryptanalyst must intercept tremendous amounts of ciphertext to be able to understand this structure. Furthermore, even when he has sufficient material, the analytical work required is much greater since the redundancy has been diffused over a large number of individual statistics.

This notion of diffusion is achieved by both the p-adic and g-adic-based systems (as has been shown in the entropy curves in the previous section). The inherent structure of the p-adic field and, more importantly, that of the g-adic ring, yields to a random spreading of the elements of $Q_{p_{AB}}$ and Q_g over the considered range. As has been shown, even with a long ciphertext, a cryptanalyst still cannot study the structure of the corresponding codes.

Another concept, also introduced by Shannon [49] and achieved in the proposed systems, is that of confusion.

Confusion attempts at making the relation between the statistics of the enciphered message and the key a very complex and involved one. This is clearer in the case of the g-adic based system where no deterministic link is apparent between p_{AB}, n and the corresponding terms generated in Q_g.

Two interleaved elements of complexity are thus achieved by this system: first, the diffusion over the g-adic numbers of the ciphertext sequence which are themselves embedded into a higher class of confusion created by the infinitely many combination possibilities of the values of p_{AB} and n.

6.3 Authentication and a New Digital Signature Procedure Based on p-adic Number Systems

In section 4.2, it was pointed out that authentication and digital signature represent an extremely important requirement in any cryptographic system.

It should be emphasized that the lack of a user authentication procedure in a system enables any intruder to generate illegal messages and send them to target recipients. However, in the proposed public-key cryptosystem, this problem is solved since A uses his own secret key to send the headers h_1 and h_2, and B uses A's public key to decipher the message based on h_1 and h_2.

Another user, X, or an intruder, cannot use K_{s_A} since it is a

secret key only known to A, and if X generated a message to B, based on his (i.e., X's) secret key (or any other random key), then B would not receive the corresponding headers properly since he has to identify X (not A) as the sender and consequently use y_X as the public deciphering key.

The other feature necessitated in an efficient cryptosystem is a secure yet adaptable signature procedure.

A signature procedure acts both ways in case of disputes. First, it is necessary to prevent any receiver, B, of a message from changing the contents of the message, or forging a message and sending it to himself, claiming that A has originated it. On the other hand, this procedure is necessary in order to confirm whether A has or has not effectively transmitted a certain message. In some cases, A may deny having sent a message and, in others, he may claim that he has sent a particular message while, in fact, he has not.

The signature procedure is thus required to solve such conflicts. It is needed to be secure such that B cannot forge it and such that it can only be revealed to a higher authority if and when the case arises. It should also be adaptable in such a way that it is message-dependent in order to either confirm or deny the contents of the ciphertext, along with identifying the message originator.

In this section a new digital signature is suggested which satisfies the above requirements. It is based on discrete logarithms and is a direct function of the message.

The method is depicted in Fig. 6.23. After the sender A has sent his final message block, he can send one extra enciphered code based on $Q_{p_{AB}}$ or Q_g. This is his signature S:

$$S \equiv (M)^{K_{s_A}} \pmod{p_{AB}} \tag{6.8}$$

where M is a function of the message elements m_i:

$$M = f(m_i) \tag{6.9}$$

This function may be, for instance, the product of all numerators and denominators constituting the message, or part of it, according to a protocol agreed upon by all users of the system.

Although the value S can be sent as an extra header at the end of the message (i.e., in integer form), it is suggested that it is preferable to send it in p-adic (or g-adic) code. In this way it will constitute an added proof for the values of p_{AB} (and n) on which the earlier message was originally based.

The adaptability of this procedure is clear since it is directly linked to the message, to p_{AB} and to n. Its security is also guaranteed since it is based on A's secret key and on discrete logarithms. Thus B cannot compute the value of K_{s_A} from a knowledge of S and M.

From a cryptanalyst's point of view, he is initially faced with the harder problem of deciphering the p-adic or g-adic code of the signature before even attempting to solve a discrete logarithm. In

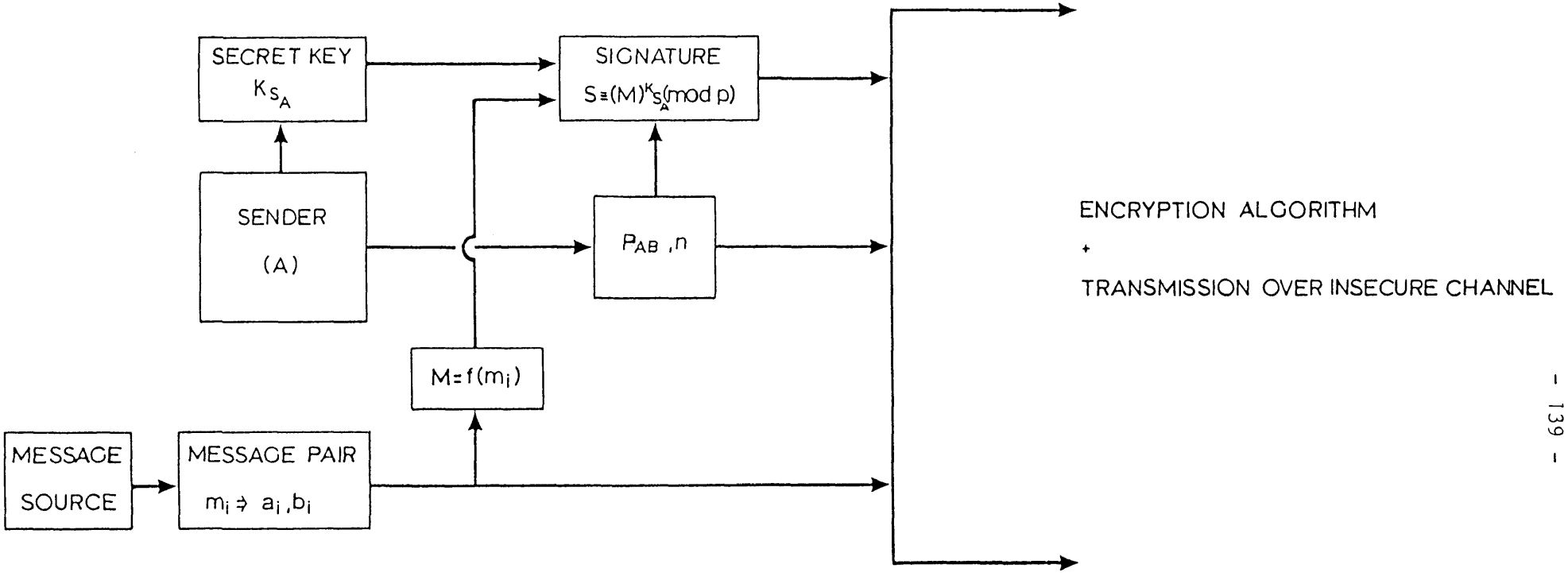


Fig. 6.23: Flow of Information in the Digital Signature Procedure

other words, for a cryptanalyst to break the digital signature procedure suggested in this section, he has to break the entire encryption algorithm described over the previous chapter.

6.4 Comparison with other Public-Key Encryption Systems

In the previous sections a detailed evaluation of the cryptographic viability of the proposed scheme was performed. This evaluation was based on all possible cryptanalytic approaches which aimed at breaking the p-adic and then the g-adic systems.

It was shown that, from the number theoretic side, breaking the system is equivalent to solving discrete logarithms over finite fields. From the statistical side, it was demonstrated that an attack on the system without a knowledge of p_{AB} and n has to consider an extremely large search space where the possibility of detecting useful information is ruled out as a possible computational undertaking by the cryptanalyst.

The suggested system reveals distinct advantages over the existing public-key systems.

First, compared with the Merkle-Hellman algorithm, it does not suffer from the need of selecting any particular sequences for the encryption process: the Merkle-Hellman algorithm was shown to be breakable under some assumptions if care was not taken while creating the encrypting vector. This vector, due to its dimension and magnitudes of its elements, proved impractical to be stored in a

public file shared by a large number of users. Also, when used for obtaining signatures, the trapdoor knapsack algorithm appears to be weak [38] in comparison with other existing public-key systems.

Secondly, compared with the Diffie-Hellman and RSA systems, the proposed system is shown to be more secure than either scheme. This is due to the fact that it uses features from both systems, namely the discrete logarithm and factorization problems respectively.

While in the Diffie-Hellman scheme the emphasis is solely on the key distribution method rather than on the encryption algorithm, the proposed system makes use of the key distribution scheme and introduces a new and secure enciphering method.

Whereas the RSA system considers sending the message directly over the channel by raising it to the power of the encryption key, the proposed system conceals the plaintext in the form of p-adic and g-adic codes thus making any cryptanalytic attack a highly complex operation.

The encryption algorithms proposed in this system, not only aim at generating a highly confusing and seemingly random ciphertext but, more importantly, they maintain a high degree of security against different attacks.

This has been shown (in section 4.4.6) not to be the case in the RSA system and which is considered to be the most practically secure cryptosystem known to the author. Although the attacks

directed at the RSA system in section 4.4.6 are not guaranteed to succeed in all cases, still they do give strong indications of the vulnerability of such a system.

The two suggested schemes, however, have been shown , through the previous analyses, to safeguard the cryptographic security of the system against directed attacks. It has been demonstrated that, given that no algorithm exists which efficiently solves the discrete logarithm problem, the p-adic and g-adic schemes are theoretically and experimentally secure and that they do not share the same degree of vulnerability as the RSA system.

The signature procedure described in the previous section is totally message and sender dependent. If it is transmitted in the form of a p-adic or g-adic sequence, as it is suggested, it forms a direct unbreakable and unforgeable link between the sender, the message he sent, p_{AB} and n.

When comparing the proposed system with other cryptosystems, it is also important to consider the bit rate required for the efficient implementation of each scheme.

In this respect, the Merkle-Hellman system requires an extremely large key size: the elements of the vector a are 0(200)-bit natural numbers and the vector x is 0(n)-binary bits where n is the length of both vectors a and x. Referring to sections 4.4.3 and 4.4.4, the generated ciphertext, c , is thus:

$$O\left(\sum_{i=1}^{n} (99 + i)\right) = O\left(\frac{n}{2} (n + 199)\right) \qquad (6.10)$$

i.e., O(15000)-bits for $n \cong 100$.

This order of magnitude is demanded in the Merkle-Hellman implementation to ensure maximum cryptographic security. Furthermore, complicated encryption algorithms to produce "safe" sets of the extremely large vector a and to store these sets in the public directory are necessarily required.

The RSA system, on the other hand, is found to be very efficient in terms of bit rate requirements. The system is considered secure if the ciphertext is O(700)-bits.

But, as has been demonstrated earlier in the thesis, the successive encryption algorithm can break the RSA system and, consequently, it has to be thoroughly investigated when using such orders of magnitude to ensure the practical safety of its use.

In the proposed cryptsystem, however, it is shown that, compared with the RSA system in particular, the ciphertext is much longer than in the RSA case. This extended length of the code may be regarded as a disadvantage in the system. However, unlike the RSA system, the proposed schemes are shown to be theoretically and practically more secure than the RSA algorithm. Longer ciphertexts do not constitute a problem in practical implementation where software

algorithms are fast and efficient. Furthermore, if the hardware implementation of the system can be achieved, then the speed and efficiency of operation will improve even more.

In conclusion, the comparison between the two systems, is, in fact, a tradeoff between security and practicality of implementation. In cryptographic systems design, the chief target is security, and this is believed to be achieved through the schemes developed in this thesis.

Also, the longer codes contribute to the inherent existence of the concepts of diffusion and confusion discussed in section 6.2. These concepts, consequently, enable the users of the system to consider smaller orders of magnitude in the ciphertext. Prevention from cryptanalytic attacks is thus based on confusion rather than on extremely large numbers. This has an important impact where mini-computers or, eventually, microcomputers are to be used in a multi-user cryptographic network.

CHAPTER 7

CONCLUSIONS

7.1 Summary of Contributions

This thesis was concerned with the application of p-adic number systems in the design of a secure public-key cryptographic scheme.

p-adic number systems, although introduced by Hensel in 1908, have only recently attracted attention for their possible uses in exact linear computations, matrix processors and signal transformations.

On the other hand, the subject of cryptography with all its pertaining features is currently being seriously investigated by researchers. The tremendous developments in the areas of computer science, design and analysis of efficient algorithms have led to the consequent development of cryptology.

Although cryptology involves many sciences which all contribute to the design of secure systems, it is fundamentally based on number theory.

It was through the study of p-adic number systems that a link was made between the two areas. Due to their inherent structure and their seemingly random distribution, these systems were shown, in this thesis, to be an important tool in the design of a public-key cryptographic system.

After a brief introduction in chapter 1 to p-adic number systems and to cryptography, in general, chapter 2 is a detailed analysis of variable-length p-adic number systems. The canonic infinite p-adic expansions corresponding to rational numbers have been discussed and the algorithm proposed by Krishnamurthy for the conversion from the rational field Q to the p-adic field Q_p, where, p is a prime, was studied.

By considering infinite p-adic expansions and by detecting the period in the recurrent elements of these expansions, a simple formula was derived which proved the existence of periodicity and which allowed the simple calculation of corresponding p-adic periods.

Although simple in form, this formula had to be computationally efficient. A practical efficient algorithm was then developed which computed the p-adic period, λ, for any rational number, given the prime p. Comparison between the periods corresponding to different denominators and their Euler totient function was also presented for different prime values.

To make use of this efficient ability to compute the p-adic period and with a view to using the p-adic number systems in designing a secure cryptographic scheme, another algorithm for the conversion from Q to Q_p wa developed and which is considered to be better than Krishnamurthy's algorithm, since this latter algorithm is non terminating. The conversion algorithm presented in section 2.4,

however, computes all the aperiodic and periodic elements in the p-adic canonic expansion and terminates after one complete cycle of the recurring elements.

In chapter 3, finite-segment p-adic number systems calculated over the finite field $\hat{Q}_p$, were studied. Through a brief mathematical introduction to these systems, it was pointed out that, for rational values to be uniquely represented in $\hat{Q}_p$, they have to satisfy certain bounds. Although these bounds were never mathematically justified, an explanation for their existence was provided in section 3.1.

Section 3.2 dealt with the structure of Hensel codes and the different representations in $\hat{Q}_p$ of different rational number structures. Then, arithmetic operations in $\hat{Q}_p$ were discussed. Krishnamurthy's algorithms for the four main operations were given. However, it is seen that these algorithms do not always generate the correct results and, in some cases, the obtained results do not correspond to any Hensel code. The algorithms which were developed in section 3.3.5 overcame these drawbacks in Krishnamurthy's algorithms and allowed closed arithmetic operations to be performed in $\hat{Q}_p$.

In section 3.4, the different methods for the inverse conversion from $\hat{Q}_p$ to Q were presented. These methods were analysed separately, and it is seen that the method of successive additions proved to be the most flexible and fastest amongst all other techniques and, hence, was the one to be adopted in the proposed cryptographic algorithm.

Finally, the limitations of using segmented p-adic number systems were discussed, in general, and particularly in connection with cryptographic usage and, consequently, it was decided to construct the cryptographic scheme based on variable-length p-adic number systems.

In chapter 4, the mathematical theory behind cryptographic systems was presented and the requirements which should be met by such systems were detailed. A conventional cryptosystem was described and the practical limitations of such a system were pointed out. This led to the introduction of public-key cryptosystems which enabled privacy, security, authentication and digital signature procedures to be incorporated in one system. The needs for authentication and digital signatures were clarified in section 4.2.

It was mentioned earlier that the design of secure cryptographic systems relied mainly on number theory. Nevertheless, other disciplines also control the efficiency of such systems. One such science is that of complexity theory. Problems were classified in complexity classes according to the degrees of difficulty in solving them. The hardest of these classes is the so called NP-complete class. Designers of cryptosystems attempt to base their algorithms on this class of complexity. The new algorithms presented in this thesis appear to belong to the class NP.

Based on all these features, the concept of public-key cryptosystems was then explained and, in section 4.4, the 3 main systems which fall into that category of cryptographic schemes were discussed in detail.

Chapter 5 is the actual implementation of p-adic number systems in the design of the proposed cryptosystem. The p-adic code structure to be used in this system was introduced and it was seen how variable-length p-adic number systems can be efficiently implemented in the proposed scheme. The conversion from Q_p to Q was discussed again based on this proposed structure. Rational values in Q would correspond to elements in the system's alphabet.

The subject of alphabet generation was then discussed and the necessary restrictions for any ambiguity-free cryptosystem were presented. Practical ways of overcoming these restrictions were also given.

Then, in section 5.4, the first scheme based on p-adic number fields was put forward and the corresponding encryption and decryption algorithms were explained. The second scheme is an extension of the first and relies on the g-adic rings, Q_g, where g is a power of a prime p_{AB} (i.e., p_{AB}^n). With one extra header in the encryption algorithm, the security of the cryptosystem is exponentially increased. However, to be able to convert the ciphertext, now in Q_g, to the rational field Q to recover the message, an algorithm was developed which first converted Q_g into $Q_{p_{AB}}$ and finally, elements in $Q_{p_{AB}}$ are converted to their rational value through the usual conversion algorithm.

The proposed system was then objectively evaluated in chapter 6. Different cryptanalysis approaches were directed, first at the p-adic scheme and, secondly, to the g-adic scheme. It was shown that

the system incorporated the two safety building blocks of the Diffie-Hellman and RSA systems. It relies on the difficulty of solving discrete logarithms over GF(p) and on the difficulty of factorizing a number into its prime components. Considering the latter feature, it is thought that the factorization problem in the proposed system is even harder to solve than in the RSA system, since it invloves the decomposition of a number into two primes or more and, also, over GF(p), which is not the case in the RSA system.

Unless any major developments take place regarding the solution of the above-mentioned problems, the system is considered theoretically safe and practically safer than either the Diffie-Hellman and RSA systems. This is because of the random configuration of the ciphertext which follows directly from the application of p-adic number systems.

To prove the randomness of this structure, statistical analysis of the distribution of these numbers in $Q_{p_{AB}}$ was performed by calculating the frequencies of occurrence in a particular range, and by computing the entropy of the p-adic numbers occurring in a message. Entropy calculations were based on random messages of lengths 100, 1000 and 10000 characters. The messages themselves were, first, constituted of 26 random alphabet characters. Then, the analysis was based on an English alphabet generated by considering the relative frequencies of the English characters. Random messages of the above lengths were also simulated and the entropy studied. It was shown that the p-adic numbers did, in fact, occur in a random manner in the ciphertext and, even though, a cryptanalyst could detect those numbers

which may not be present in the ciphertext, this deduced information does not help the cryptanalyst in solving the problem he is faced with in any way.

The randomness in the distribution of p-adic numbers has also been tested in the case of specific English messages. Two messages were considered and values of p_{AB} = 2909, 4001 and 9967 were used to study the frequencies of occurrence of the p-adic numbers in the corrresponding ciphertext.

One consideration which was then dealt with was the periodicity parameters. It was shown that, although the p-adic numbers are randomly spread over a particular range, the periodicity parameters, on the other hand, tend to be small in magnitude (relative to the p-adic numbers) and repetitive. This leads to the possibility of detecting the code boundaries. And, although this fact does not constitute a risk to the proposed system, an algorithm was developed to embed the periodicity parameters in the code. Numbers occurring in the code, now, have the added feature of being spread over GF(p) instead of $Q_{p_{AB}}$.

The same approach was carried over to the second system. Only there, the cryptanalysis task proved to be more complex. A cryptanalyst, in the g-adic system, is faced with a search which increases exponentially with the increase of the power n. It is claimed that this search falls in the NP-class of complexity problems unless otherwise proved.

The concepts of diffusion and confusion introduced by Shannon were also discussed in regard to the two schemes.

In section 6.3, a new digital signature method was suggested which relied on discrete logarithms and p-adic number systems. In it, the user would sign his messages and it guaranteed the authenticity of the message contents in a transmitted ciphertext. The receiver could not initiate messages and claim they were transmitted by a certain sender and he could not change the contents of a received message. A sender could not deny having sent a message either. The signature procedure suggested links the sender, the message, p_{AB} and n in one unbreakable unit.

Finally, a comparison was carried out, in section 6.4, between the proposed system and already existing cryptographic algorithms. According to the analysis carried out in this thesis, it was found that, unless an algorithm for solving discrete logarithms over GF(p) was developed, the p-adic and, more so, the g-adic system prove to be safer than the Diffie-Hellman and RSA systems. Breaking the two proposed schemes is equivalent to breaking the Diffie-Hellman system.

7.2 Suggestions for Future Research

In this section, some ideas are suggested for further investigation. These ideas relate to both the analysis performed in this thesis and to cryptography in general.

First, regarding the analysis of the p-adic and g-adic schemes, it was shown that these schemes were secure given any cryptanalytic attack. This statement is made, at least, from the author's point of view. Methods and ways of cryptanalysis other than those reported in this thesis can be thought of and directed to the schemes. Once the method has withstood all attacks for a sufficient length of time, it may be practically used with a reasonable amount of confidence.

The number theoretic algorithms which are at the core of the proposed system should be further investigated. Attempts of breaking the system may lead to efficient algorithms for solving the discrete logarithm problem over finite fields or the factorization problem, both knowingly unsolved as yet.

From a practical side, it is suggested that these schemes be implemented on microprocessors for the speed and flexibility of operation. This would involve the implementation of p-adic arithmetic on microprocessors, a very attractive project in its own right, since it would lead to extremely fast and error-free computations. On the other hand, it would involve a thorough study of finite-segment p-adic number systems and a further investigation of the closure of arithmetic operations in $\hat{Q}_p$.

Regarding cryptography in general, security measures should be studied to safeguard the information held in the public directory.

If a cryptographic system is to be of value, its keys must be protected. In some situations, the "loss" of cryptographic keys may occur. This may be due to:

a) hardware malfunction

b) software error

c) human error in handling the keys

The effects of losing a key should be studied and techniques of overcoming such mishaps should be investigated.

Finally, dedicated hardware for the implementation of cryptographic schemes is needed. For instance, the computation involved in finding suitable prime numbers is heavy; exponential functions involved run too slowly when carried out by software; modular arithmetic is not yet efficiently implemented. All these factors determine the need for hardware designs of cryptosystems which will run in a fast and efficient way.

REFERENCES

[1] W.W. Adams and L.J. Goldstein
"Introduction to Number Theory"
Prentice-Hall, Inc., 1976.

[2] L. Adleman
"A Subexponential Algorithm for the Discrete Logarithm Problem with Applications to Cryptography"
Proc. of the 20th IEEE Symp. on Foundations of Computer Science, October 1979, pp. 55-60.

[3] E. Alparslan
"Finite p-adic Computing Systems with Possible Applications"
Ph.D. Dissertation, Dept. of Elec. Eng., University of Maryland, College Park, 1975.

[4] Y. Amice
"Les Nombres p-adiques"
Presses Universitaires de France, 1975.

[5] T.A. Apostol
"Introduction to Analytic Number Theory"
Springer-Verlag, 1976.

[6] G. Bachman
"Introduction to p-adic Numbers and Valuation Theory"
Academic Press, Inc., 1964.

[7] H. Beker and F. Piper
"Cipher Systems - The Protection of Communications"
Northwood Publications, 1982.

[8] H. Beker and F. Piper
"Communications Security; a Survey of Cryptography"
IEEE Proc., vol. 129, pt. A, No. 6, August 1982, pp. 357-376.

[9] Z.I. Borevich and I.R. Shafarevich
"Number Theory"
Academic Press, 3rd Printing, 1973.

[10] B. Bosworth
"Codes, Ciphers and Computers - An Introduction to Information Security"
Hayden Book Company, Inc., 1982.

[11] D.W. Davies and W.L. Price
"The Applications of Digital Signatures Based on Public-Key Cryptosystems"
Proc. of the 5th ICCC, October 1980, pp. 525-530.

[12] D.W. Davies and W.L. Price
"Security for Computer Networks"
John Wiley and Sons, 1984.

[13] W. Diffie and M.E. Hellman
"New Directions in Cryptography"
IEEE Trans. on Inf. Th., vol. IT-22, No. 6, November 1976, pp.644-654.

[14] W. Diffie and M.E. Hellman
"Privacy and Authentication: an Introduction to Cryptography"
Proc. of the IEEE, vol.67, No.3, March 1979, pp. 397-427.

[15] M.R. Garey and D.S. Johnson
"Computers and Intractability - A Guide to the Theory of NP-Completeness"
W.H. Freeman and Co., 1979.

[16] R.N. Gorgui-Naguib and R.A. King
"Comments and Corrections on 'Matrix Processors Using p-adic Arithmetic for Exact Linear Computations'"
To be published, IEEE Trans. on Computers.

[17] R.N. Gorgui-Naguib and A. Leboyer
"Comment on 'Determination of p-adic Transform Bases and Lengths'"
Electronics Letters, vol. 21, No.20, September 1985, pp.905-906.

[18] R.T. Gregory and E.V. Krishnamurthy
"Methods and Applications of Error-Free Computation"
Texts and Monographs in Computer Science, Springer-Verlag, 1984.

[19] R.K. Guy
"Unsolved Problems in Number Theory"
Problem Books in Mathematics, vol.1, Springer-Verlag, 1981.

[20] M.E. Hellman
"The mathematics of Public-Key Cryptography"
Scientific American, August 1979, pp. 130-139.

[21] K. Hensel
"Theorie der Algebraischen Zahlen"
Teubner, Leipzig, 1908.

[22] K. Hensel
"Zahlentheorie"
Göschen, Berlin and Leipzig, 1913.

[23] T. Herlestam
"Critical Remarks on Some Public-Key Cryptosystems"
BIT, vol. 18, 1978, pp. 493-496.

[24] D. Kahn
"The Codebreakers, the Story of Secret Writing"
New York, Macmillan, 1967.

[25] D.E. Knuth
"The Art of Computer Programming - vol. II: Seminumerical Algorithms"
Addison-Wesley Publishing Co., 1969.

[26] D.E. Knuth
"The Art of Computer Programming - vol. III: Sorting and Searching"
Addison-Wesley Publishing Co., 1973.

[27] N. Koblitz
"p-adic Numbers, p-adic Analysis and Zeta-Functions"
Graduate Texts in Mathematics, Springer-Verlag,1977.

[28] A.G. Konheim
"Cryptography, a Primer"
John Wiley and Sons, 1981.

[29] E.V. Krishnamurthy, T. Mahadeva Rao and K. Subramanian
"Finite-Segment p-adic Number Systems with Applications to Exact Computation"
Proc. Indian Acad. Sci., vol. 81A, No. 2, 1975, pp. 58-79.

[30] E.V. Krishnamurthy, T. Mahadeva Rao and K. Subramanian
"p-adic Arithmetic Procedures for Exact Matrix Computations"
Proc. Indian Acad. Sci., vol. 82A, No. 5, 1975, pp. 165-175.

[31] E.V. Krishnamurthy
"Matrix Processors Using p-adic Arithmetic for Exact Linear Computations"
IEEE Trans. on Computers, vol. C-26, No. 7, July 1977, pp. 633-639.

[32] A. Leboyer
"p-adic Numbers-p-adic Transform"
MSc. Communications Report, Imperial College, June 1985.

[33] W.J. Leveque
"Fundamentals of Number Theory"
Addison-Wesley Publishing Co., 1977.

[34] V. Loahakosol and W. Surakampontorn
"p-adic Transforms"
Electronics Letters, vol. 20, No. 18, August 1984, pp. 726-727.

[35] K. Mahler
"p-adic Numbers and their Functions"
Cambridge Tracts in Mathematics, Cambridge University Press, 2nd ed., 1981.

[36] R.C. Merkle
"Secure Communications Over Insecure Channels"
Communications of the ACM, vol. 21, No. 4, APril 1978, pp. 294-299.

[37] R.C. Merkle
"Secrecy, Authentication and Public Key Systems"
Ph.D. Dissertation, Dept. of Elect. Eng., Stanford Univ., Stanford, CA, June 1979.

[38] R.C. Merkle and M.E. Hellman
"Hiding Information and Signatures in Trapdoor Knapsacks"
IEEE Trans. on Inf. Th., vol. IT-24, No. 5, September, 1978, pp. 525-530.

[39] N.M. Nasrabadi
"Orthogonal Transforms and their Applications to Image Coding"
Ph.D. Thesis, Imperial College, London, 1984.

[40] N.M. Nasrabadi and R.A. King
"Fast Digital Convolution Using p-adic Transforms"
Electronics Letters, vol. 19, No. 7, March 1983, pp. 266-267.

[41] N.M. Nasrabadi and R.A. King
"Complex Number Theoretic Transform in p-adic Field"
Proc. of IEEE-ICASSP 84, 1984, pp.28A.4.1 - 28A.4.3.

[42] I. Niven and H.S. Zuckerman
"An Introduction to the Theory of Numbers"
John Wiley and Sons, 4th ed., 1980.

[43] S.-C. Pei and J.-L. Wu
"Determination of p-adic Transform Bases and Lengths"
Electronics Letters, vol. 21, 1985, pp. 431-432.

[44] S.C. Pohlig
"Algebraic and Combinatoric Aspects of Cryptography"
Ph.D. Dissertation, Dept. of Elect. Eng., Stanford Univ., Stanford, CA, June 1977.

[45] S.C. Pohlig and M.E. Hellman
"An Improved Algorithm for computing Logarithms Over GF(p) and its Crypographic Significance"
IEEE Trans. on Inf. Th., vol. IT-24, No.1, January 1978, pp. 106-110.

[46] R.L. Rivest, A. Shamir and L. Adleman
"A Method for Obtaining Digital Signature and Public-Key Cryptosystems"
Communications of the ACM, vol.21, No.2, February 1978, pp.120-126.

[47] M.R. Schroeder
"Number Theory in Science and Communication"
Springer Series in Information Sciences, Springer-Verlag, 1984.

[48] A. Shamir and R.E. Zippel
"On the Security of the Merkle-Hellman Cryptographic Scheme"
IEEE Trans. on Inf. Th., vol. IT-26, No.3, May 1980, pp.339-340.

[49] C.E. Shannon
"Communication Theory of Secrecy Systems"
Bell Sys. Tech. J., 28, 1949, pp.657-715.

[50] C.E. Shannon
"Prediction and Entropy of Printed English"
Bell Sys. Tech. J., 30, 1951, pp.50-64.

[51] G.J. Simmons and M.J. Norris
"Preliminary Comments on the M.I.T. Public-Key Cryptosystem"
Cryptologia, vol.1, No.4, October 1977, pp.406-414.

APPENDIX A

Table of $H(p,r,\alpha)$ Codes for $p = 5$, $r = 4$

b \ a	1	2	3	4	5	6
1	.1000	.2000	.3000	.4000	.0100	.1100
2	.3222	.1000	.4222	.2000	.0322	.3000
3	.2313	.4131	.1000	.3313	.0231	.2000
4	.4333	.3222	.2111	.1000	.0433	.4222
5	1.000	2.000	3.000	4.000	.1000	1.100
6	.1404	.2313	.3222	.4131	.0140	.1000
7	.3302	.1214	.4021	.2423	.0330	.3142
8	.2414	.4333	.1303	.3222	.0241	.2111
9	.4201	.3012	.2313	.1124	.0420	.4131
10	3.222	1.000	4.222	2.000	.3222	3.000
11	.1332	.2120	.3403	.4240	.0133	.1411
12	.3424	.1404	.4333	.2313	.0342	.3222
13	.2034	.4014	.1143	.3123	.0203	.2232
14	.4101	.3302	.2013	.1214	.0410	.4021
15	2.313	4.131	1.000	3.313	.2313	2.000
16	.1234	.2414	.3104	.4333	.0123	.1303
17	.3043	.1132	.4121	.2210	.0304	.3342

b \ a	7	8	9	10	11	12
1	.2100	.3100	.4100	.0200	.1200	.2200
2	.1322	.4000	.2322	.0100	.3322	.1100
3	.4313	.1231	.3000	.0413	.2231	.4000
4	.3111	.2000	.1433	.0322	.4111	.3000
5	2.100	3.100	4.100	.2000	1.200	2.200
6	.2404	.3313	.4222	.0231	.1140	.2000
7	.1000	.4302	.2214	.0121	.3423	.1330
8	.4030	.1000	.3414	.0433	.2303	.4222
9	.3432	.2243	.1000	.0301	.4012	.3313
10	1.322	4.000	2.322	.1000	3.322	1.100
11	.2204	.3041	.4324	.0212	.1000	.2332
12	.1202	.4131	.2111	.0140	.3020	.1000
13	.4212	.1341	.3321	.0401	.2430	.4410
14	.3222	.2423	.1134	.0330	.4431	.3142
15	4.313	1.231	3.000	.4131	2.231	4.000
16	.2042	.3222	.4402	.0241	.1421	.2111
17	.1431	.4420	.2024	.0113	.3102	.1240

b \ a	13	14	15	16	17
1	.3200	.4200	.0300	.1300	.2300
2	.4322	.2100	.0422	.3100	.1422
3	.1413	.3231	.0100	.2413	.4231
4	.2433	.1322	.0211	.4000	.3433
5	3.200	4.200	.3000	1.300	2.300
6	.3404	.4313	.0322	.1231	.2140
7	.4142	.2000	.0402	.3214	.1121
8	.1241	.3111	.0130	.2000	.4414
9	.2124	.1420	.0231	.4432	.3243
10	4.322	2.100	.4222	3.100	1.422
11	.3120	.4403	.0340	.1133	.2411
12	.4424	.2404	.0433	.3313	.1342
13	.1000	.3034	.0114	.2143	.4123
14	.2343	.1000	.0201	.4302	.3013
15	1.413	3.231	.1000	2.413	4.231
16	.3340	.4030	.0310	.1000	.2234
17	.4234	.2323	.0412	.3401	.1000

APPENDIX B

Table of Variable-Length p-adic Codes for p = 5 and $\gamma = 17$

b \ a	1	2	3
1	1 2 10	1 2 20	1 2 30
2	1 2 32	1 2 10	1 2 42
3	2 2 231	2 2 413	1 2 10
4	1 2 43	1 2 32	1 2 21
5	1 2 10	1 2 20	1 2 30
6	2 2 140	2 2 231	1 2 32
7	6 2 3302142	6 2 1214230	6 2 4021423
8	2 2 241	1 2 43	2 2 130
9	6 2 4201243	6 2 3012432	2 2 231
10	1 2 32	1 2 10	1 2 42
11	5 2 133240	5 2 212041	5 2 340332
12	2 2 342	2 2 140	1 2 43
13	4 2 20341	4 2 40143	4 2 11430
14	6 2 4101343	6 2 3302142	6 2 2013431
15	2 2 231	2 2 413	1 2 10
16	4 2 12340	2 2 241	4 2 31042
17	16 2 30431210240132342	16 2 11323420431210240	16 2 41210240132342043

b \ a	4	5	6
1	1 2 40	1 3 010	1 3 110
2	1 2 20	1 3 032	1 2 30
3	2 2 331	2 3 0231	1 2 20
4	1 2 10	1 3 043	1 2 42
5	1 2 40	1 2 10	1 3 110
6	2 2 413	2 3 0140	1 2 10
7	6 2 2423021	6 3 03302142	6 2 3142302
8	1 2 32	2 3 0241	1 2 21
9	6 2 1124320	6 3 04201243	2 2 413
10	1 2 20	1 2 32	1 2 30
11	5 2 424033	5 3 0133240	5 2 141120
12	2 2 231	2 3 0342	1 2 32
13	4 2 31232	4 3 020341	4 2 22321
14	6 2 1214230	6 3 04101343	6 2 4021423
15	2 2 331	2 2 231	1 2 20
16	1 2 43	4 3 012340	2 2 130
17	16 2 22102401323420431	16 3 030431210240132342	16 2 33420431210240132

b \ a	7	8	9
1	1 3 210	1 3 310	1 3 410
2	1 3 132	1 2 40	1 3 232
3	2 2 431	2 3 1231	1 2 30
4	1 2 31	1 2 20	1 3 143
5	1 3 210	1 3 310	1 3 410
6	2 2 240	2 2 331	1 2 42
7	1 2 10	6 2 4302142	6 2 2214230
8	2 2 403	1 2 10	2 2 341
9	6 2 3432012	6 2 2243201	1 2 10
10	1 3 132	1 2 40	1 3 232
11	5 2 220411	5 2 304112	5 2 432403
12	2 2 120	2 2 413	1 2 21
13	4 2 42123	4 2 13410	4 2 33212
14	1 2 32	6 2 2423021	6 2 1134310
15	2 2 431	2 3 1231	1 2 30
16	4 2 20421	1 2 32	4 2 44023
17	16 2 14312102401323420	16 2 44204312102401323	16 2 20240132342043121

b \ a	10	11	12
1	1 3 020	1 3 120	1 3 220
2	1 3 010	1 3 332	1 3 110
3	2 3 0413	2 3 2231	1 2 40
4	1 3 032	1 2 41	1 2 30
5	1 2 20	1 3 120	1 3 220
6	2 3 0231	2 3 1140	1 2 20
7	6 3 01214230	6 2 3423021	6 3 13302142
8	1 3 043	2 2 230	1 2 42
9	6 3 03012432	6 2 4012432	2 2 331
10	1 2 10	1 3 332	1 3 110
11	5 3 0212041	1 2 10	5 2 233240
12	2 3 0140	2 2 302	1 2 10
13	4 3 040143	4 2 24301	4 2 44103
14	6 3 03302142	6 2 4431013	6 2 3142302
15	2 2 413	2 3 2231	1 2 40
16	2 3 0241	4 2 14210	1 2 21
17	16 3 011323420431210240	16 2 31024013234204312	16 2 12401323420431210

b \ a	13	14	15
1	1 3 320	1 3 420	1 3 030
2	1 3 432	1 3 210	1 3 042
3	2 3 1413	2 3 3231	1 3 010
4	1 3 243	1 3 132	1 3 021
5	1 3 320	1 3 420	1 2 30
6	2 2 340	2 2 431	1 3 032
7	6 2 4142302	1 2 20	6 3 04021423
8	2 3 1241	1 2 31	2 3 0130
9	6 2 2124320	6 3 14201243	2 3 0231
10	1 3 432	1 3 210	1 2 42
11	5 2 312041	5 2 440332	5 3 0340332
12	2 2 442	2 2 240	1 3 043
13	1 2 10	4 2 30341	4 3 011430
14	6 2 2343101	1 2 10	6 3 02013431
15	2 3 1413	2 3 3231	1 2 10
16	4 2 33402	2 2 403	4 3 031042
17	16 2 42342043121024013	16 2 23234204312102401	16 3 0412102401323420 43

b \ a	16	17
1	1 3 130	1 3 230
2	1 3 310	1 3 142
3	2 3 2413	2 3 4231
4	1 2 40	1 3 343
5	1 3 130	1 3 230
6	2 3 1231	2 3 2140
7	6 2 3214230	6 3 11214230
8	1 2 20	2 2 441
9	6 2 4432012	6 2 3243201
10	1 3 310	1 3 142
11	5 3 1133240	5 2 241120
12	2 2 331	2 3 1342
13	4 2 21430	4 2 41232
14	6 2 4302142	6 2 3013431
15	2 3 2413	2 3 4231
16	1 2 10	4 2 22340
17	16 2 34013234204312102	1 2 10

APPENDIX C1

Program for the Variable-Length p-adic/Rational Conversion

```
C       FINITE REPRESENTATION OF INFINITE p-ADIC EXPANSIONS
C
C         RATIONAL / INFINITE p-ADIC CONVERSION BASED ON
C         ---------- THE PERIOD CALCULATION ----------
C                    ----------------------
C
C
C
C
C ------------------------------------------------------------------
C NOTE: THE STRUCTURE OF THE RESULTING CODE IN THIS PROGRAM FOLLOWS
C       THE THEORETICAL DESCRIPTION.
C ------------------------------------------------------------------
C
C
C
C
C
C Main program calling different subroutines from the menu
C
        INTEGER P
C
C Reading the initial value of the prime p of the infinite p-adic field.
C This value can be altered during execution through the command RDA.
C
        CALL CREAD(P)
C
10      WRITE (5,999)
999     FORMAT (/,2X,'Enter command or HLP')
        READ (5,998) COM
998     FORMAT (A3)
C
C These are the command routines which, in turn, will call the corresponding
C operation routines.
C
        IF (COM.EQ.'HLP'.OR.COM.EQ.'hlp') CALL CHLP
        IF (COM.EQ.'RDA'.OR.COM.EQ.'rda') CALL CREAD(P)
        IF (COM.EQ.'FOR'.OR.COM.EQ.'for') CALL CFOR(P)
        IF (COM.EQ.'INV'.OR.COM.EQ.'inv') CALL CINV(P)
        IF (COM.EQ.'END'.OR.COM.EQ.'end') GO TO 20
        GO TO 10
C
C
C
20      STOP
        END
```

```
C                      ******************
C                      *                *
C                      * COMMAND modules *
C                      *                *
C                      ******************
C
C
      SUBROUTINE CHLP
C
C Command routine to display menu items
C
      WRITE (5,*) ' Commands are:'
      WRITE (5,*) '   RDA - read a new value of P'
      WRITE (5,*) '   FOR - conversion from rational to infinite p-adic'
      WRITE (5,*) '   INV - conversion from infinite p-adic to rational'
      WRITE (5,*) '   END - end execution'
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CREAD(P)
C
C Command routine to read and pass new values of p.
C
      INTEGER STORE(50),P,SIZE
C
10    WRITE (5,999)
999   FORMAT (2X,'Enter PRIME')
      READ (5,*) P
C
C Test of primality on p
C
      I = P
      CALL PRMDIV(I,STORE,SIZE)
      IF (STORE(1).NE.P) THEN
         WRITE (5,998)
998      FORMAT (2X,'THE VALUE OF PRIME ENTERED IS NOT A PRIME')
         GO TO 10
      ENDIF
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CFOR(P)
C
C Command routine calling the 'FORWARD', rational ---> p-adic conversion
C routine, FINREP, and then printing the resulting finite representation
C of the infinite expansion
C
      INTEGER PCODE(50),P,NUM,DEN
C
      WRITE (5,999)
999   FORMAT (2X,'Enter NUM and DEN of rational')
      READ (5,*) NUM,DEN
```

```
        CALL FINREP(NUM,DEN,P,PCODE,LENGTH)
        CALL PRINT(PCODE,LENGTH)
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE CINV(P)
C
C Command routine calling the 'INVERSE', p-adic ---> rational conversion
C routine, INV.
C
        INTEGER PCODE(50),P
C
        WRITE (5,999)
999     FORMAT (2X,'Enter the order N of the Farey sequence')
        READ (5,*) N
C
        WRITE (5,998)
998     FORMAT (2X,'Enter infinite P-ADIC CODE')
        READ (5,*) PCODE(1),PCODE(2),(PCODE(I),I=3,PCODE(1)+PCODE(2)+1)
C
C L and L1 correspond to the p-adic period and to the first recurrent
C element respectively
C
        L  = PCODE(1)
        L1 = PCODE(2)
C
        CALL INV(P,PCODE,L,L1,N)
C
        RETURN
        END
```

```
C               ************************
C               *                      *
C               * INPUT/OUTPUT modules *
C               *                      *
C               ************************
C
C
      SUBROUTINE PRINT(PCODE,LENGTH)
C
C Routine to print the finite representation of variable length, LENGTH,
C of the infinite p-adic expansion
C
      INTEGER PCODE(50)
C
      WRITE (5,999)
999   FORMAT (1X,'FINITE p-ADIC REPRESENTATION OF THE EXPANSION:')
      WRITE (5,998) (PCODE(I),I=1,LENGTH)
998   FORMAT (1X,50I3)
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE RATOUT(NUM,DEN)
C
C Same routine as in program HENSEL
C
      INTEGER NUM,DEN
C
      IN = 2
      IF (IABS(NUM).GT.9) IN = 3
      IF (IABS(NUM).GT.99) IN = 4
      IF (IABS(NUM).GT.999) IN = 5
      IF (IABS(NUM).GT.9999) IN = 6
C
      ID = 1
      IF (DEN.GT.9) ID = 2
      IF (DEN.GT.99) ID = 3
      IF (DEN.GT.999) ID = 4
      IF (DEN.GT.9999) ID = 5
C
      WRITE (5,999)
999   FORMAT (1X,'RATIONAL EQUIVALENT:')
      WRITE (5,998) NUM,DEN
998   FORMAT (1X,I<IN>,'/',I<ID>)
C
      RETURN
      END
```

```
C               ***********************************************
C               *                                             *
C               * RATIONAL to p-ADIC CODE CONVERSION routine  *
C               *            and related routines             *
C               *                                             *
C               ***********************************************
C
C
      SUBROUTINE FINREP(A,B,P,PADIC,SIZE)
C
C This routine implements the theoretical algorithm developed for a
C finite representation of an infinite p-adic expansion, based on
C the p-adic period computation
C
      INTEGER PADIC(50),R(50),A,B,P,D,SIZE,PERIOD
C
C Initialization
C
      DO 10 I=1,50
         R(I) = 0
         PADIC(I) = 0
10    CONTINUE
C
C Check if (A,B)=1 and, if not, then set the updated values of A and B
C
      N1 = IABS(A)
      N2 = IABS(B)
      CALL GCD(N1,N2)
      B = N2
      IF (A.LT.0) THEN
         A = -N1
                ELSE
         A = N1
      ENDIF
C
C Check the divisibility of the denominator in ALFA. B becomes D.
C
      CALL DIVDEN(B,P,D)
C
C Calculation of ITEMP which corresponds to r in the theoretical
C development and which sets an upper bound on the computation
C
      M = MAX(IABS(A),IABS(D))
      ALOG1 = ALOG10((2.*((FLOAT(M))**2.)) + 1.)
      ALOG2 = ALOG10(FLOAT(P))
      TEMP  = ALOG1 / ALOG2
      ITEMP = IFIX(TEMP + 1.)
      IF (MOD(ITEMP,2).NE.0) ITEMP=ITEMP+1
C
C Computation of the p-adic period and putting its value in the 1st
C location
C
      L = PERIOD(P,D)
      PADIC(1) = L
C
C Implementation of the developed algorithm. INVD is the multiplicative
C inverse of the denominator D.
C
      CALL SOLVE(D,INVD,P)
```

```
        R(1) = A
        PADIC(3) = MODF((INVD*R(1)),P)
C
        DO 20 I=1,L+ITEMP-1
           R(I+1) = (R(I) - PADIC(I+2)*D) / P
           PADIC(I+3) = MODF((INVD*R(I+1)),P)
20      CONTINUE
C
C Computation of the pointer to the 1st recurrent element and putting
C this value in the 2nd location
C
        DO 30 I=2,L+ITEMP
           DO 40 J=1,I-1
              IF (R(I).EQ.R(J)) THEN
                 L1 = J
                 PADIC(2) = L1
                 GO TO 50
              ENDIF
40         CONTINUE
30      CONTINUE
C
50      SIZE = L + L1 + 1
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE GCD(A,B)
C
C Same routine as in HENSEL program
C
        INTEGER A,B,GCDVAL
C
        N1 = A
        N2 = B
        IF (N1.GT.N2) GO TO 10
        K1 = N1
        N1 = N2
        N2 = K1
10      J  = MOD(N1,N2)
        IF (J.EQ.0) GO TO 20
        N1 = N2
        N2 = J
        GO TO 10
20      GCDVAL = N2
        A  = A / GCDVAL
        B  = B / GCDVAL
C
        RETURN
        END
C
C ---------- O ----------
```

```
      SUBROUTINE DIVDEN(VIN,P,VOUT)
C
C This subroutine checks the divisibility of an integer VIN by
C a prime P. The final indivisible value, VOUT, is returned.
C
      INTEGER VIN,P,VOUT,TEMP
C
      TEMP = VIN
10    MODP = MODF(TEMP,P)
      IF (MODP.EQ.0) GO TO 20
      VOUT = TEMP
      GO TO 30
C
20    TEMP = TEMP / P
      GO TO 10
C
30    RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE SOLVE(VIN,VOUT,P)
C
C Same routine as in HENSEL program
C
      INTEGER VIN,VOUT,VALUE,P
C
      DO 10 VOUT=1,P-1
         VALUE = VIN * VOUT
         MODP  = MODF(VALUE,P)
         IF (MODP.EQ.1) GO TO 20
10    CONTINUE
C
20    RETURN
      END
C
C ---------- O ----------
C
      FUNCTION MODF(VIN,P)
C
C Same function as in HENSEL program
C
      INTEGER VIN,P
C
      MODF = MOD(VIN,P)
10    IF (MODF.LT.0) MODF=MODF+P
      IF (MODF.LT.0) GO TO 10
C
      RETURN
      END
C
C ---------- O ----------
```

```
      INTEGER FUNCTION PERIOD(P,B)
C
C Function to compute the p-adic periodicity given the rational denominator
C B and the prime p. The computation is a direct implementation of the
C theoretical algorithm developed.
C       PSTORE is a 1-d array where the odd locations contain the primes
C              and the even locations contain their corresponding powers.
C       NSTORE is a 1-d array where the locations contain the values of
C              PSTORE(i)**PSTORE(i+1)
C       FSTORE is a 2-d 'matrix' where a row is allocated to each FIi and
C              the column locations contain each prime factor of FIi raised
C              to its power.
C
      INTEGER NSTORE(50),PSTORE(100),FSTORE(50,100)
      INTEGER P,B,D,DD,Q,PSIZE,TEMP,QUOTNT,RES,FI
C
C For D=1 or D=2 (note: here, D=B), the period is always 1.
C
      IF (B.EQ.1.OR.B.EQ.2) THEN
         PERIOD = 1
         GO TO 400
      ENDIF
C
C Array initialisation
C
      DO 100 I=1,50
         NSTORE(I) = 0
100   CONTINUE
C
      DO 200 I=1,50
         DO 300 J=1,100
            FSTORE(I,J) = 0
300      CONTINUE
200   CONTINUE
C
      D  = B
      DD = D
C
C Call the prime power factorization (PPF) procedure to perform the
C PPF of the denominator D. Values are stored in PSTORE then,
C subsequently, in NSTORE, after raising the primes to their powers.
C
      CALL PRMFAC(DD,PSTORE,PSIZE)
C
      J = 1
      I = 1
      K = I + 1
10    NSTORE(J) = PSTORE(I)**PSTORE(K)
      IF (K.EQ.PSIZE) GO TO 20
      I = I + 2
      K = I + 1
      J = J + 1
      GO TO 10
```

```
C Store FI(Pi**EPSILONi) [Refer to nomenclature in the theoretical
C analysis].
C
20      DO 30 I=1,J
           TEMP = NSTORE(I)
           NSTORE(I) = FI(TEMP)
30      CONTINUE
C
        MAXSIZ = 1
C
C In the following, FSTORE is filled according to FI
C
        DO 40 I=1,J
           IF (NSTORE(I).GT.1) THEN
              TEMP = NSTORE(I)
              CALL PRMFAC(TEMP,PSTORE,PSIZE)
              FSTORE(I,1) = PSTORE(1)
              IF (PSIZE.EQ.2) GO TO 40
              INDEX = 0
              DO 50 K=3,PSIZE-1,2
                 INDEX = INDEX + 1
                 FSTORE(I,K-INDEX) = PSTORE(K)
                 IF ((K-INDEX).GT.MAXSIZ) MAXSIZ=K-INDEX
50            CONTINUE
           ENDIF
40      CONTINUE
C
C Assign the largest value of FI to PERIOD
C
        PERIOD = NSTORE(1)
        IF (J.GT.1) THEN
           DO 60 I=2,J
              IF (NSTORE(I).GT.PERIOD) PERIOD=NSTORE(I)
60         CONTINUE
        ENDIF
C
C Compute the smallest value of PERIOD which satisfies the congruence
C                    P**PERIOD = 1 (mod D)
C
        DO 70 I=1,J
           IF (NSTORE(I).GT.1) THEN
              DO 80 K=MAXSIZ,1,-1
                 IF (FSTORE(I,K).EQ.0) GO TO 80
                 QUOTNT = NSTORE(I) / FSTORE(I,K)
                 RES = MODULO(P,QUOTNT,D)
                 IF ((RES.EQ.1).AND.(QUOTNT.LE.PERIOD)) THEN
                    PERIOD = QUOTNT
                    GO TO 70
                 ENDIF
80            CONTINUE
           ENDIF
70      CONTINUE
```

```
90      IF (MOD(QUOTNT,2).EQ.0) THEN.
           QUOTNT = QUOTNT / 2
           RES    = MODULO(P,QUOTNT,D)
           IF (RES.EQ.1) THEN
                         PERIOD = QUOTNT
                         GO TO 90
           ENDIF
        ENDIF
C
C
400     RETURN
        END
C
C ---------- O ----------
C
        FUNCTION MODULO(X,N,P)
C
C Function to compute MOD (X**N,P) iteratively, for large values of X**N
C such that no overflow errors occur
C
        INTEGER X,P,COUNT
C
        MODULO = X
        COUNT  = 1
10      MODULO = MOD (MODULO*X,P)
        COUNT  = COUNT + 1
        IF (COUNT.NE.N) GO TO 10
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE PRMFAC(N,PSTORE,PSIZE)
C
        INTEGER   NSTORE(50),PSTORE(100),FLAG,POWER,PSIZE
C
C This subroutine calls the PRIME DIVISORS subroutine.
C The result is an array, NSTORE, of the prime divisors of N.
C Then, NSTORE is rearranged into another array, PSTORE, where the prime
C divisors are written into the odd-numbered locations of PSTORE and their
C corresponding powers in the adjacent even locations. Array PSTORE, so
C arranged, is the output of the subroutine.
C       FLAG is an indicator of the size, NSIZE, of NSTORE
C       POWER is the corresponding prime power
C
C Array initialization
C
        DO 10 I=1,50
          NSTORE(I) = 0
10      CONTINUE
C
        DO 20 I=1,100
          PSTORE(I) = 0
20      CONTINUE
C
        CALL PRMDIV(N,NSTORE,NSIZE)
```

```
C Initialization
C
        FLAG = 0
        I    = 1
        J    = 1
        POWER= 1
C
C Check if NSIZE=1 (i.e., N is a prime)
C
        IF (NSIZE.EQ.1) THEN
           PSTORE(J) = NSTORE(I)
           PSTORE(J+1) = 1
           GO TO 60
        ENDIF
C
30      K    = I + 1
        IF (K.EQ.NSIZE) FLAG = 1
        IF (NSTORE(I).EQ.NSTORE(K)) THEN
           PSTORE(J) = NSTORE(I)
           POWER     = POWER + 1
           IF (FLAG.EQ.1) GO TO 40
           I         = K
           GO TO 30
                                    ELSE
           PSTORE(J) = NSTORE(I)
           PSTORE(J+1) = POWER
           J = J + 2
           POWER = 1
           IF (FLAG.EQ.1) GO TO 50
           I = K
           GO TO 30
        ENDIF
C
50      PSTORE(J) = NSTORE(K)
40      PSTORE(J+1) = POWER
C
60      PSIZE = J + 1
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE PRMDIV(I,STORE,SIZE)
C
C The algorithm for finding the prime divisors of a composite integer
C is based on Eratosthenes sieve. The integer I is factored into its
C prime divisors and these prime divisors are written in ascending
C order in array STORE. SIZE is the actual size of STORE. If I is a
C prime, it is retuned as it is, with SIZE=1
C
        INTEGER STORE(50),ROOT,SIZE
C
C Initialization
C
        DO 10 SIZE = 1,50
           STORE(SIZE) = 0
10      CONTINUE
```

```
        SIZE = 0
C
30      SIZE = SIZE + 1
        ROOT = IFIX(SQRT(ABS(FLOAT(I))))
        DO 20 L = 2,ROOT
           K = MOD(I,L)
           IF (K.NE.0) GO TO 20
           STORE(SIZE) = L
           NEXT = I/L
           I = NEXT
           GO TO 30
20      CONTINUE
C
        STORE(SIZE) = I
C
50      RETURN
        END
C
C ---------- O ----------
C
        INTEGER FUNCTION FI(N)
C
C This function computes the Euler Totient Function of any integer N.
C
        FI = 1
        DO 10 I=2,N-1
           N1 = N
           N2 = I
20         J  = MOD(N1,N2)
           IF (J.EQ.0) GO TO 10
           IF (J.EQ.1) THEN
              FI = FI + 1
              GO TO 10
           ENDIF
           N1 = N2
           N2 = J
           GO TO 20
10      CONTINUE
C
        RETURN
        END
```

```
C               ***********************************************
C               *                                             *
C               * p-ADIC CODE to RATIONAL CONVERSION routine  *
C               *             and related routines            *
C               *                                             *
C               ***********************************************
C
C
      SUBROUTINE INV(P,PADIC,L,L1,N)
C
C This routine is an implementation of the theoretical algorithm developed.
C Given the order, N,  of the Farey sequence (which may correspond to GAMA
C in the cryptographic alphabet), the rational equivalent of an infinite
C p-adic expansion (having a finite p-adic code representation) is computed
C and printed out
C
      INTEGER PADIC(50),PADICT(50),P,R
C
      ALOG1 = ALOG10((2.*((FLOAT(N))**2.))+1.)
      ALOG2 = ALOG10(FLOAT(P))
      REALR = ALOG1 / ALOG2
      R     = IFIX(REALR+1.)
      IF (MOD(R,2).NE.0) R=R+1
      IF (R.GT.(L+L1-1)) THEN
         INDEX = L+L1+2
         DO 10 I=INDEX,R+2
            PADIC(I) = PADIC(I-L)
10       CONTINUE
      ENDIF
C
      DO 20 I=1,R
         PADICT(I) = PADIC(I+2)
20    CONTINUE
C
      CALL CONVRT(PADICT,P,R,NUM,DEN)
      CALL RATOUT(NUM,DEN)
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CONVRT(PCODE,P,R,NUM,DEN)
C
C Same routine as in HENSEL program but with minor modifications to
C the array structures
C
      INTEGER PCODE(50),PCODET(50),P,R,NUM,DEN,ACOUNT,NEGTV,C,CARRY
C
      ACOUNT = 0
      NUM    = 0
      DEN    = 0
      NEGTV  = 0
C
      DO 10 I=1,R
         PCODET(I) = PCODE(I)
10    CONTINUE
```

```
70      DO 20 I=(R/2)+1,R
           IF (PCODE(I).NE.0) GO TO 30
20      CONTINUE
        GO TO 60
C
30      DO 40 I=(R/2)+1,R
           IF (PCODE(I).NE.(P-1)) GO TO 50
40      CONTINUE
        NEGTV = 1
        GO TO 60
C
50      C = 0
        DO 80 I=1,R
           PCODE(I) = PCODE(I)+PCODET(I)+C
           C = CARRY(PCODE(I),P)
           PCODE(I) = MOD(PCODE(I),P)
80      CONTINUE
C
        ACOUNT = ACOUNT + 1
        GO TO 70
C
60      DEN = ACOUNT + 1
        IF (NEGTV.EQ.0) GO TO 90
        CALL PCOMP(PCODE,P,R,PCODE)
C
90      DO 100 I=1,R/2
           NUM = NUM+PCODE(I)*(P**(I-1))
100     CONTINUE
C
        IF (NEGTV.EQ.1) NUM=-NUM
C
        RETURN
        END
C
C ---------- O ----------
C
        INTEGER FUNCTION CARRY(VIN,P)
C
C Same function as in HENSEL program
C
        INTEGER VIN,P,VTEMP,C
C
        C = 0
        VTEMP = VIN
20      IF (VTEMP.LT.P) GO TO 10
        VTEMP = VTEMP - P
        C = C + 1
        GO TO 20
10      CARRY = C
C
        RETURN
        END
C
C ---------- O ----------
```

```
      SUBROUTINE PCOMP(PCODE,P,R,PBAR)
C
C Same routine as in HENSEL program but with minor modifications to
C the array structures
C
      INTEGER PCODE(50),PBAR(50),P,R
C
      PBAR(1) = P - PCODE(1)
      DO 10 I=2,R
         PBAR(I) = P - (PCODE(I)+1)
10    CONTINUE
C
      RETURN
      END
```

APPENDIX C2

Program for the Finite-Segment p-adic Conversion and Full Arithmetic Package

```
C                RATIONAL / HENSEL CODE CONVERSION
C                             AND
C             FINITE-SEGMENT p-ADIC ARITHMETIC PACKAGE
C             ----------------------------------------
C
C
C
C
C
C ---------------------------------------------------------------------
C NOTE: THIS PROGRAM IS DESIGNED SUCH THAT THE p-ADIC NUMBERS IN THE
C       HENSEL CODES LIE IN THE RANGE 0 TO 9. IF THIS RANGE IS TO BE
C       EXCEEDED, MODIFICATIONS SHOULD BE DONE TO THE I/O MODULES.
C ---------------------------------------------------------------------
C
C
C
C
C Main program calling different subroutines from the menu
C
C
        INTEGER PRIME,RANGE
C
C Reading initial values of the prime, p, and the range, r, of the
C segmented p-adic field. These values can be altered during execution
C through the command RDA.
C
        CALL CREAD(PRIME,RANGE)
C
10      WRITE (5,999)
999     FORMAT (2X,'Enter command or HLP')
        READ (5,998) COM
998     FORMAT (A3)
C
C These are the command routines which, in turn will call the corresponding
C operation routines.
C
        IF (COM.EQ.'HLP'.OR.COM.EQ.'hlp') CALL CHLP
        IF (COM.EQ.'RDA'.OR.COM.EQ.'rda') CALL CREAD(PRIME,RANGE)
        IF (COM.EQ.'FOR'.OR.COM.EQ.'for') CALL CFOR (PRIME,RANGE)
        IF (COM.EQ.'INV'.OR.COM.EQ.'inv') CALL CINV (PRIME,RANGE)
        IF (COM.EQ.'CMP'.OR.COM.EQ.'cmp') CALL CCMP (PRIME,RANGE)
        IF (COM.EQ.'ADD'.OR.COM.EQ.'add') CALL CADD (PRIME,RANGE)
        IF (COM.EQ.'SUB'.OR.COM.EQ.'sub') CALL CSUB (PRIME,RANGE)
        IF (COM.EQ.'MUL'.OR.COM.EQ.'mul') CALL CMUL (PRIME,RANGE)
        IF (COM.EQ.'DIV'.OR.COM.EQ.'div') CALL CDIV (PRIME,RANGE)
        IF (COM.EQ.'END'.OR.COM.EQ.'end') GO TO 20
        GO TO 10
C
C
C
20      STOP
        END
```

```
C                     ********************
C                     *                  *
C                     * COMMAND modules  *
C                     *                  *
C                     ********************
C
C
        SUBROUTINE CREAD(P,R)
C
C Command routine to read and pass new values of P and R in H(P,R)
C
        INTEGER STORE(10)
        INTEGER P,R,SIZE
C
10      WRITE (5,999)
999     FORMAT (2X,'Enter PRIME and RANGE')
        READ (5,*) P,R
C
C Test of primality on p and check that r is even
C
        I = P
        DO 20 SIZE=1,10
           STORE(SIZE) = 0
20      CONTINUE
C
        SIZE = 0
40      SIZE = SIZE + 1
        ROOT = IFIX(SQRT(FLOAT(I)))
        DO 30 J=2,ROOT
           K = MOD(I,J)
           IF (K.NE.0) GO TO 30
           STORE(SIZE) = J
           NEXT = I / J
           I = NEXT
           GO TO 40
30      CONTINUE
        STORE(SIZE) = I
C
        IF (STORE(1).NE.P) THEN
           WRITE (5,998)
998        FORMAT (2X,'THE VALUE OF PRIME ENTERED IS NOT A PRIME')
           GO TO 10
        ENDIF
C
        IF (MOD(R,2).NE.0) THEN
           WRITE (5,997)
997        FORMAT (2X,'RANGE MUST BE EVEN')
           GO TO 10
        ENDIF
C
        RETURN
        END
C
C ---------- O ----------
```

```
      SUBROUTINE CFOR(P,R)
C
C Command routine calling the 'FORWARD', rational ---> Hensel code conversion
C routine, HCODE, and then printing the resulting Hensel code.
C
      INTEGER PCODE(0:20)
      INTEGER P,R,NUM,DEN
C
      WRITE (5,999)
999   FORMAT (2X,'Enter NUM and DEN of rational')
      READ (5,*) NUM,DEN
C
C Test that the values of NUM and DEN are within Krishnamurthy's bounds
C
      N = INT(SQRT((FLOAT(P)**FLOAT(R)-1.)/2.))
      IF (NUM.GT.N.OR.DEN.GT.N) THEN
         WRITE (5,998) N
998      FORMAT (2X,'WARNING - VALUES ARE OUTSIDE THE BOUND:  +/-',I4)
      ENDIF
C
      CALL HCODE(NUM,DEN,P,R,PCODE)
C
      CALL PRINT(PCODE,R)
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CINV(P,R)
C
C Command routine calling the 'INVERSE', Hensel code ---> rational,
C conversion routine, CONVRT, and printing the equivalent rational number.
C It uses subroutine RDCODE to read a Hensel code from the terminal.
C
      INTEGER PCODE(0:20)
      INTEGER P,R,NUM,DEN
C
      WRITE (5,999)
999   FORMAT (2X,'Enter Hensel code')
C
      CALL RDCODE (PCODE)
      CALL CONVRT(PCODE,P,R,NUM,DEN)
      CALL RATOUT(NUM,DEN)
C
      RETURN
      END
C
C ---------- O ----------
```

```
      SUBROUTINE CCMP(P,R)
C
C Command routine calling the complementation routine, HCOMP, after some
C pre-processing on the Hensel code read in. The pre-processing is necessary
C to pass the code to HCOMP without the p-adic point. The command module,
C however, places the p-adic point back in its position prior to the output
C stage.
C       PLOC is the location of the p-adic point
C
      INTEGER PCODE(0:20),PTEMP(0:20),PCOMP(0:20)
      INTEGER P,R,PLOC
C
C Reading a p-adic code using RDCODE.
C
      WRITE (5,999)
999   FORMAT (2X,'Enter Hensel code to be complemented')
      CALL RDCODE(PCODE)
C
C Location of the p-adic point
C
      DO 10 I = 0, R
         IF (PCODE(I).EQ.-1) PLOC = I
10    CONTINUE
C
C Duplicating PCODE in PTEMP without the p-adic point
C
      IF (PLOC.EQ.0) GO TO 20
      DO 30 I = 0, PLOC-1
         PTEMP(I) = PCODE(I)
30    CONTINUE
20    DO 40 I = PLOC, R-1
         PTEMP(I) = PCODE(I+1)
40    CONTINUE
C
C Calling the complementation routine. The result is put back in PTEMP.
C
      CALL HCOMP(PTEMP,P,R,PTEMP)
C
C Replacing the p-adic point back in its position. The complete p-adic
C complement is written onto PCOMP.
      PCOMP(PLOC) = -1
      IF (PLOC.EQ.0) GO TO 50
      DO 60 I = 0, PLOC-1
         PCOMP(I) = PTEMP(I)
60    CONTINUE
50    DO 70 I = PLOC+1, R
         PCOMP(I) = PTEMP(I-1)
70    CONTINUE
C
      WRITE (5,998)
998   FORMAT (1X, 'COMPLEMENT:')
      CALL PRINT (PCOMP,R)
C
      RETURN
      END
C
C ---------- O ----------
```

```
      SUBROUTINE CADD(P,R)
C
C Command routine calling the addition routine. It reads two rational numbers,
C converts them into their Hensel codes and then performs the segmented p-adic
C addition. The result is output in Hensel code and rational forms.
C
      INTEGER PCODE1(0:20),PCODE2(0:20),PCODEA(0:20)
      INTEGER NUM1,DEN1,NUM2,DEN2,N1,D1,N2,D2,P,R,NUM,DEN
C
      WRITE (5,999)
999   FORMAT (2X,'Enter NUM and DEN of 1st rational')
      READ (5,*) NUM1,DEN1
C
      WRITE (5,998)
998   FORMAT (2X,'Enter NUM and DEN of 2nd rational')
      READ (5,*) NUM2,DEN2
C
      N1 = NUM1
      D1 = DEN1
      N2 = NUM2
      D2 = DEN2
C
      CALL HCODE(N1,D1,P,R,PCODE1)
      CALL HCODE(N2,D2,P,R,PCODE2)
C
C The last parameter, 1, in HADD corresponds to the addition routine flag:
C               1 --> routine may be executed twice
C               0 --> routine executed once only
C This parameter is important in case we have leading 0's.
C
      CALL HADD(NUM1,DEN1,NUM2,DEN2,PCODE1,PCODE2,PCODEA,P,R,1)
C
      WRITE (5,997)
997   FORMAT (1X,'ADDITION RESULT:')
      CALL PRINT(PCODEA,R)
      WRITE (5,996)
996   FORMAT (1X,'or')
      CALL CONVRT(PCODEA,P,R,NUM,DEN)
C
      CALL RATOUT(NUM,DEN)
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CSUB(P,R)
C
C Command routine to perform the segmented p-adic subtraction. The
C subtraction process eventually, is exactly similar to the addition.
C This command routine is mainly concerned with sending a message re.
C the format of the input data.
C
      INTEGER P,R
```

```
        WRITE (5,999)
999     FORMAT (2X,'In the following, assign a minus sign to the rational ',
     +             'to be subtracted:')
C
C From now on it is exactly like CADD
C
        CALL CADD(P,R)
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE CMUL(P,R)
C
C Command routine calling the multiplication subroutine, HMULT. It reads
C two rational numbers, +ve or -ve, converts them into their Hensel codes
C and then performs the segmented p-adic multiplication. The product is
C output in Hensel code and rational forms.
C
        INTEGER PCODE1(0:20),PCODE2(0:20),PCODEM(0:20)
        INTEGER NUM1,DEN1,NUM2,DEN2,N1,D1,N2,D2,NUM,DEN,P,R
C
        WRITE (5,999)
999     FORMAT (2X, 'Enter NUM and DEN of 1st rational')
        READ (5,*) NUM1,DEN1
C
        WRITE (5,998)
998     FORMAT (2X, 'Enter NUM and DEN of 2nd rational')
        READ (5,*) NUM2,DEN2
C
        N1 = NUM1
        D1 = DEN1
        N2 = NUM2
        D2 = DEN2
C
        CALL HCODE(N1,D1,P,R,PCODE1)
        CALL HCODE(N2,D2,P,R,PCODE2)
        CALL HMULT(NUM1,DEN1,NUM2,DEN2,PCODE1,PCODE2,P,R,PCODEM)
C
        WRITE (5,997)
997     FORMAT (1X, 'MULTIPLICATION PRODUCT:')
        CALL PRINT(PCODEM,R)
        WRITE (5,996)
996     FORMAT (1X, 'or')
        CALL CONVRT(PCODEM,P,R,NUM,DEN)
        CALL RATOUT (NUM,DEN)
C
        RETURN
        END
C
C ---------- O ----------
```

```
      SUBROUTINE CDIV(P,R)
C
C Command routine calling the division subroutine, HDIV.It reads two
C rational numbers, +ve or -ve, converts them into their Hensel codes
C and then performs the segmented p-adic division. The quotient is
C output in Hensel code and rational forms.
C
      INTEGER PCODE1(0:20),PCODE2(0:20),PCODED(0:20)
      INTEGER NUM1,DEN1,NUM2,DEN2,N1,D1,N2,D2,NUM,DEN,P,R
C
      WRITE (5,999)
999   FORMAT (2X,'Enter NUM and DEN of dividend')
      READ (5,*) NUM1,DEN1
C
      WRITE (5,998)
998   FORMAT (2X,'Enter NUM and DEN of divisor')
      READ (5,*) NUM2,DEN2
C
      N1 = NUM1
      D1 = DEN1
      N2 = NUM2
      D2 = DEN2
C
      CALL HCODE(N1,D1,P,R,PCODE1)
      CALL HCODE(N2,D2,P,R,PCODE2)
      CALL HDIV(NUM1,DEN1,NUM2,DEN2,PCODE1,PCODE2,P,R,PCODED)
C
      WRITE (5,997)
997   FORMAT (1X,'DIVISION QUOTIENT:')
      CALL PRINT(PCODED,R)
      WRITE (5,996)
996   FORMAT (1X,'or')
      CALL CONVRT(PCODED,P,R,NUM,DEN)
      CALL RATOUT(NUM,DEN)
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE CHLP
C
C Command routine to display all the commands menu for novice users
C
      WRITE (5,*) ' commands are :'
      WRITE (5,*) '   RDA - read new values of P and R'
      WRITE (5,*) '   FOR - conversion from RATIONAL to HENSEL CODE'
      WRITE (5,*) '   INV - conversion from HENSEL CODE TO RATIONAL'
      WRITE (5,*) '   CMP - complement of a Hensel code'
      WRITE (5,*) '   ADD - segmented p-adic addition of two rationals'
      WRITE (5,*) '   SUB - segmented p-adic subtraction of two rationals'
      WRITE (5,*) '   MUL - segmented p-adic multiplication of two rationals'
      WRITE (5,*) '   DIV - segmented p-adic division of two rationals'
      WRITE (5,*) '   END - end execution'
C
      RETURN
      END
```

```
C                     ************************
C                     *                      *
C                     * INPUT/OUTPUT modules *
C                     *                      *
C                     ************************
C
C
      SUBROUTINE RDCODE(CODE)
C
C Routine to read a variable-length Hensel code. It is self formatting
C according to the number of characters, NCH, read in.
C
      LOGICAL*1 IN(0:20)
      INTEGER CODE(0:20)
      INTEGER RANGE
C
      READ (5,999) NCH,(IN(I),I=1,NCH)
999   FORMAT (Q,22A1)
C
C Having read an alphanumeric string (the p-adic point and the code digits),
C the following converts the p-adic point to -1.
C
      RANGE = NCH - 1
      DO 10 I = 0, RANGE
         CODE(I) = IN(I+1) - π060
         IF (IN(I+1).EQ.'.') CODE(I) = -1
10    CONTINUE
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE PRINT(PCODE,R)
C
C Routine to print a Hensel code of length R
C
      LOGICAL*1 OUT(0:20)
      INTEGER PCODE(0:20)
      INTEGER R
C
C Since the p-adic point is treated as -1 in all the arithmetic, here the
C -1 in the PCODE is replace by a dot in OUT. π060 + PCODE(I) returns the
C octal value of PCODE(I), and the corresponding final format is alphanumeric.
C
      DO 10 I = 0, R
         OUT(I) = π060 + PCODE(I)
         IF (PCODE(I).EQ.-1) OUT(I) = '.'
10    CONTINUE
```

```
        WRITE (5,999)
999     FORMAT (1X,'HENSEL CODE:')
        WRITE (5,998) (OUT(I), I=0,R)
998     FORMAT (1X,<R+1>A1,//)
C
        RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE RATOUT(NUM,DEN)
C
C Routine to print any rational number in the form A/B.
C Note that 1 extra location is reserved in IN to provide for the possible
C -ve sign.
C
        INTEGER NUM,DEN
C
        IN = 2
        IF (IABS(NUM).GT.9) IN = 3
        IF (IABS(NUM).GT.99) IN = 4
        IF (IABS(NUM).GT.999) IN = 5
        IF (IABS(NUM).GT.9999) IN = 6
C
        ID = 1
        IF (DEN.GT.9) ID = 2
        IF (DEN.GT.99) ID = 3
        IF (DEN.GT.999) ID = 4
        IF (DEN.GT.9999) ID = 5
C
        WRITE (5,999)
999     FORMAT (1X,'RATIONAL EQUIVALENT:')
        WRITE (5,998) NUM,DEN
998     FORMAT (1X,I<IN>,'/',I<ID>,//)
C
        RETURN
        END
```

```
C               ***********************************************
C               *                                             *
C               * RATIONAL to HENSEL CODE CONVERSION routine  *
C               *            and related routines             *
C               *                                             *
C               ***********************************************
C
C
      SUBROUTINE HCODE(A,B,P,R,PADIC)
C
C given a rational number ALFA=A/B with (A,B)=1 and B.NE.0, its
C p-adic expansion is obtained through this subroutine.
C In the following analysis PADICT is a temporary array storing the
C actual integer elements of the Hensel code, while PADIC is the
C final array with the p-adic point represented as -1.
C
      INTEGER PADICT(20),PADIC(0:20)
      INTEGER A,B,C,D,BNEW,NUANPN,DEANPN,CGAMA,DGAMA,
     +        P,R,X
C
C Checking if (A,B) = 1 and, if not, then get the updated values of A and B.
C
      CALL GCD(A,B)
C
C Initialization of the p-adic index.
C
      J = 0
C
C STEP 1: Set C = the numerator in ALFA.
C         Check the divisibility of the denominator in ALFA
C         by the prime P. This will control the position of
C         the p-adic point.
C         Set D = the denominator in ALFA (or the  new  one
C         if divisible by P).
C
      C = A
      CALL DIVDEN(B,P,BNEW,N)
      D = BNEW
C
20    J = J + 1
C
C STEP 2: solve the congruence DX = 1 (mod P).
C         if X(j) is a solution, then PADIC(j) = C.X(j) (mod P).
C
      CALL SOLVE (D,X,P)
      PADICT(J) = MODF (C*X, P)
C
C Check if we have reached the required range R.
C
      IF (J.EQ.R) GO TO 30
C
C STEP 3: set GAMA = (C/D) - PADIC(j).
C
      NUANPN = PADICT(J)
      DEANPN = 1
      CALL RATSUB (C,D,NUANPN,DEANPN,CGAMA,DGAMA)
```

```
C STEP 4: Set the new value of D = DGAMA.
C         The corresponding value (CGAMA) of C, however,
C         will be divisible by P.
C         Set C = CGAMA / (P**1).
C         Go to STEP 2 after increasing the p-adic index by 1.
C
        D = DGAMA
        CALL DIVNUM(CGAMA,P,C)
        GO TO 20
C
C Transfering the Hensel code from PADICT to PADIC with the p-adic
C point in its final position.
C
30      IF (N.EQ.0) GO TO 40
        PADIC(N) = -1
        DO 50 I = 0, N-1
           PADIC(I) = PADICT(I+1)
50      CONTINUE
        DO 60 I = N+1, R
           PADIC(I) = PADICT(I)
60      CONTINUE
        GO TO 80
40      PADIC(N) = -1
        DO 70 I = N+1, R
           PADIC(I) = PADICT(I)
70      CONTINUE
C
80      RETURN
        END
C
C ---------- O ----------
C
        SUBROUTINE GCD(A,B)
C
C Subroutine checking if (A,B) = 1. If not, then A and B are reduced
C such that A -> A/GCD   and   B -> B/GCD.
C
        INTEGER A,B,GCDVAL
C
        N1 = A
        N2 = B
        IF (N1.GT.N2) GO TO 10
        K1 = N1
        N1 = N2
        N2 = K1
10      J  = MOD(N1,N2)
        IF (J.EQ.0) GO TO 20
        N1 = N2
        N2 = J
        GO TO 10
20      GCDVAL = N2
        A  = A / GCDVAL
        B  = B / GCDVAL
C
        RETURN
        END
C
C ---------- O ----------
```

```
      SUBROUTINE DIVNUM (VIN,P,VOUT)
C
C subroutine for the divisibility of the numerator by P
C such that if ALFA=A/B, then A = C * P
C VIN  is the value corresponding to A
C VOUT is the value corresponding to C
C
      INTEGER VIN,VOUT,P
C
      MODP = MODF(VIN, P)
      IF (MODP.EQ.0) GO TO 10
      VOUT = VIN
      GO TO 20
C
10    VOUT = VIN / P
C
20    RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE DIVDEN (VIN,P,VOUT,N)
C
C Subroutine checking the divisibility of the denominator by P,
C such that if ALFA=A/B, then B = D * (P**N).
C VIN  is the value corresponding to B
C VOUT is the value corresponding to D
C Unlike DIVNUM ]H, we are interested in the powers of P in this case,
C because these will control the position of the p-adic point.
C
      INTEGER VIN,VOUT,P
C
      N = 0
10    MODP = MODF(VIN,P)
      IF (MODP.EQ.0) GO TO 20
      VOUT = VIN
      GO TO 30
C
20    N = N + 1
      VIN = VIN / P
      GO TO 10
C
30    RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE SOLVE (VIN,VOUT,P)
C
C Subroutine to solve the congruence DX = 1 (mod P) in STEP 2
C of the conversion algorithm.
C VIN  is the value corresponding to D
C VOUT is the value corresponding to X
C
      INTEGER VIN,VOUT,VALUE,P
```

```
      DO 10 VOUT = 1, P-1
         VALUE = VIN * VOUT
         MODP  = MODF(VALUE, P)
         IF (MODP.EQ.1) GO TO 20
10    CONTINUE
C
20    RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE RATSUB(NUM1,DEN1,NUM2,DEN2,NUM,DEN)
C
C Subroutine to subtract 2 fractions and return the resulting value
C in fraction form (corresponding to NUM and DEN).
C
      INTEGER NUM1,NUM2,NUM,DEN1,DEN2,DEN
C
      IF (DEN1.EQ.DEN2) GO TO 10
      DEN = DEN1 * DEN2
      NUM = (NUM1 * DEN2) - (NUM2*DEN1)
      GO TO 20
C
10    DEN = DEN1
      NUM = NUM1 - NUM2
C
20    RETURN
      END
C
C ---------- O ----------
C
      FUNCTION MODF(VIN,P)
C
C Function returning MOD(VIN,P) with the only difference that -ve
C values of MOD, which may be obtained through the usual intrinsic
C function MOD, are accounted for with their equivalent +ve values.
C
      INTEGER VIN,P
C
      MODF = MOD(VIN,P)
10    IF (MODF.LT.0) MODF = MODF + P
      IF (MODF.LT.0) GO TO 10
C
      RETURN
      END
```

```
C                  ****************************************
C                  *                                      *
C                  * Segmented p-adic ADDITION routine    *
C                  *       and related routines           *
C                  *                                      *
C                  ****************************************
C
C
      SUBROUTINE HADD(NUM1,DEN1,NUM2,DEN2,HCODE1,HCODE2,HCODEA,
     +                P,R,ADFLAG)
C
C Subroutine performing the addition operation in the segmented p-adic
C field Qp of length R. It uses the updated addition algorithm developed
C by Gorgui-Naguib and King. The result of HCODE1 + HCODE2 is stored in
C HCODEA. The subroutine makes use of subroutine RSHIFT to shift right
C either HCODE1 or HCODE2 to align the p-adic point before adding, and
C of function CARRY to calculate carries arising from subsequent additions.
C Notice that the addition process is done over a Hensel code size of:
C              SIZE = NEWR
C where NEWR is the updated range size due to a right-shift of one of the
C codes (a p-adic point misalignment).
C ADFLAG is a parameter = 0 or 1 for disabling or enabling the multiple
C (at most one more time) running of this subroutine. Initially ADFLAG is
C set to 1 so that, in case leading 0's exist, the routine may be run once
C more over an extended range (which is returned as REXT from subroutine
C LEAD0)
C
      INTEGER HCODE1(0:20),HCODE2(0:20),HCODEA(0:20),
     +        HCD1S(0:20),HCD2S(0:20)
C
C PCOUNT is a counter for the number of digits (not necessarily 0's) preceding
C        the p-adic point.
C PLOCK  locks onto the above value of PCOUNT for further analysis of these
C        digits.
C
      INTEGER NUM1,DEN1,NUM2,DEN2,P,R,ADFLAG,PCOUNT,PLOCK,C,CARRY
C
C Initialize PCOUNT and carry
C
      PCOUNT = 0
      C      = 0
C
      DO 10 I = 0, R
         IF (HCODE1(I).EQ.-1) N1=I
         IF (HCODE2(I).EQ.-1) N2=I
10    CONTINUE
C
      IF (N1.GT.N2) GO TO 20
      IF (N2.GT.N1) GO TO 40
      NEWR = R
      GO TO 60
```

```
C Shift right according to obtained values of N1 and N2
C
20      NEWR = R + N1 - N2
        CALL RSHIFT(HCODE2,HCD2S,N1-N2,NEWR)
        DO 30 I = 0, NEWR
           HCODE2(I) = HCD2S(I)
30      CONTINUE
        GO TO 60
C
40      NEWR = R + N2 - N1
        CALL RSHIFT(HCODE1,HCD1S,N2-N1,NEWR)
        DO 50 I = 0, NEWR
           HCODE1(I) = HCD1S(I)
50      CONTINUE
C
C At this stage the p-adic point is aligned and the addition process is
C carried out from left to right.
C Note that NEWR, which is a function of R, may be implicitely increased
C according to the number of leading 0's found during the initial run of
C the subroutine (i.e. if LEAD0 is to be called during the first run).
C
60      DO 70 I = 0, NEWR
           IF (HCODE1(I).EQ.-1) GO TO 80
           PCOUNT = PCOUNT + 1
           HCODEA(I) = HCODE1(I) + HCODE2(I) + C
           C = CARRY(HCODEA(I),P)
           HCODEA(I) = MOD(HCODEA(I),P)
           GO TO 70
80         PLOCK = PCOUNT
           HCODEA(I) = -1
70      CONTINUE
C
C Check if any digits precede the p-adic point.
C
        IF (PLOCK.EQ.0) GO TO 90
C
C If so, and ADFLAG is enabled, then call LEAD0 to check for leading 0's.
C
        IF (ADFLAG.EQ.1) CALL LEAD0(NUM1,DEN1,NUM2,DEN2,HCODEA,P,R,PLOCK)
C
C Here, a final check on 0's left of the p-adic point is performed.
C If the 'most significant bit' is 0, then the whole addition result is
C 'shifted left' by 1 location until a non-zero digit is met. Note that the
C location of the p-adic point retains its relative location throughout the
C shifting process (i.e., it is also shifted along with the other digits).
C a part of the addition code).
C
90      IF (HCODEA(0).NE.0) GO TO 100
        CALL LSHIFT(HCODEA,HCODEA,1,NEWR)
        GO TO 90
C
100     RETURN
        END
C
C ---------- 0 ----------
```

```
      SUBROUTINE RSHIFT (SEQIN,SEQOUT,K,M)
C
C This subroutine shifts a given sequence SEQIN of length M by K places
C to the right and fills-in the gaps with 0's. The resulting shifted
C sequence is passed as SEQOUT.
C
      INTEGER SEQIN(0:20),SEQOUT(0:20)
C
      DO 10 I = 0, K-1
         SEQOUT(I) = 0
10    CONTINUE
C
      DO 20 I = K, M
         SEQOUT(I) = SEQIN(I-K)
20    CONTINUE
C
      RETURN
      END
C
C ---------- O ----------
C
      INTEGER FUNCTION CARRY(VIN,P)
C
C Function returning the carry arising from a value VIN w.r.t. P
C
      INTEGER VIN,VTEMP,P,C
C
      C = 0
      VTEMP = VIN
20    IF (VTEMP.LT.P) GO TO 10
      VTEMP = VTEMP - P
      C     = C + 1
      GO TO 20
10    CARRY = C
C
      RETURN
      END
C
C ---------- O ----------
C
      SUBROUTINE LSHIFT(SEQIN,SEQOUT,K,M)
C
C This subroutine shifts left a given sequence SEQIN of length M by K places
C to the left and fills-in the gaps with 0's. The resulting shifted sequence
C is passed as SEQOUT.
C
      INTEGER SEQIN(0:20),SEQOUT(0:20)
C
      DO 10 I = 0, M-K
         SEQOUT(I) = SEQIN(I+K)
10    CONTINUE
C
      DO 20 I = M-K+1, M
         SEQOUT(I) = 0
20    CONTINUE
C
      RETURN
      END
C
C ---------- O ----------
```

```
      SUBROUTINE LEAD0(NUM1,DEN1,NUM2,DEN2,HCODEA,P,R,PLOCK)
C
C This subroutine checks if some or all the digits preceding the p-adic
C point = 0. The number of leading 0's is given by ZCOUNT.
C Then, the new Hensel codes of the rationals are obtained over a range REXT
C such that REXT = R + ZCOUNT and is evenμ (this is also why NUM1, DEN1,
C NUM2 and DEN2 were passed all through the program).
C Then, the addition routine HADD is called again and is performed over this
C range.
C
      INTEGER HCODEA(0:20),PCODE1(0:20),PCODE2(0:20)
      INTEGER NUM1,DEN1,NUM2,DEN2,P,R,PLOCK,ZCOUNT,REXT
C
C Initialize ZCOUNT
C
      ZCOUNT = 0
C
      DO 10 I = 1, PLOCK
         IF (HCODEA(I-1).EQ.0) ZCOUNT=ZCOUNT+1
         IF (HCODEA(I)  .NE.0) GO TO 20
10    CONTINUE
C
20    REXT = R + ZCOUNT
C
C REXT must be even
C
      IF ((MOD(REXT,2)).NE.0) REXT=REXT+1
C
C Extended Hensel codes over REXT
C
      CALL HCODE(NUM1,DEN1,P,REXT,PCODE1)
      CALL HCODE(NUM2,DEN2,P,REXT,PCODE2)
C
C Perform extended addition. Here, ADFLAG is passed as 0 to disable any
C further execution of HADD.
C
      CALL HADD(NUM1,DEN1,NUM2,DEN2,PCODE1,PCODE2,HCODEA,P,REXT,0)
C
      RETURN
      END
```

```
C               ********************************************
C               *                                          *
C               * Segmented p-adic COMPLEMENTATION routine *
C               *                                          *
C               ********************************************
C
C
      SUBROUTINE HCOMP(HCODE,P,R,HBAR)
C
C Subroutine to complement a Hensel code, HCODE, and put the result
C in HBAR.
C NOTE: HCODE is assumed free of the p-adic point (-1) and so is HBAR
C       (That is why the range is R-1 and not R).
C
      INTEGER HCODE(0:20),HBAR(0:20)
      INTEGER P,R
C
      HBAR(0) = P - HCODE(0)
      DO 10 I = 1, R-1
         HBAR(I) = P - (HCODE(I) + 1)
10    CONTINUE
C
      RETURN
      END
C
C
C
C               ********************************************
C               *                                          *
C               * Segmented p-adic MULTIPLICATION routine  *
C               *           and related routines           *
C               *                                          *
C               ********************************************
C
C
      SUBROUTINE HMULT(NUM1,DEN1,NUM2,DEN2,HCODE1,HCODE2,P,R,HCODEM)
C
C Subroutine performing the multiplication operation in the segmented p-adic
C field Qp of length R. It uses the updated multiplication algorithm developed
C by Gorgui-Naguib and King. The product of HCODE1 * HCODE2 is stored in
C HCODEM.
C The subroutine operates on codes devoid of the p-adic point and it restitutes
C the p-adic point in the final product according to:
C               E(gama) = E(alfa) + E(beta) + k + 1
C where (k+1) denotes the amount of left-shift of one of the operand codes
C due to existing leading 0's and misalignment of the p-adic points in both
C codes.
C Subroutine CHK0 is the subroutine responsible for checking on the existence
C of such leading 0's and their removal prior to the multiplication process
```

```
C       PLOC1 is the loaction of the p-adic point in HCODE1
C       PLOC2 . . .       . . .     .     . HCODE2
C       PLOC  . . .       . . .     .     . the final product HCODEM
C       KZERO . . number of leadind 0's in any, and only, one of the codes
C       CA    . . carry arising from addition
C       CM    . . .     .       .    multiplication
C
        INTEGER HCODE1(0:20),HCODE2(0:20),HCODEM(0:20),
     +          HTEMP1(0:20),HTEMP2(0:20),HTEMPM(0:20),
     +          HSHIFT(0:20)
        INTEGER NUM1,DEN1,NUM2,DEN2,P,R,PLOC1,PLOC2,PLOC,KZERO,
     +          CA,CM,CARRY
C
C Initialization.
C
        KZERO = 0
        CA    = 0
        CM    = 0
        DO 5 I = 0, 20
           HTEMPM(I) = 0
5       CONTINUE
C
C Locate relative positions of p-adic points
C
        DO 10 I = 0, R
           IF (HCODE1(I).EQ.-1) PLOC1 = I
           IF (HCODE2(I).EQ.-1) PLOC2 = I
10      CONTINUE
C
C The following conditions govern the necessity for checking on leading 0's
C in any one (and only one) of the operand codes and also for rewriting
C HCODE1 and HCODE2 without their p-adic point in HTEMP1 and HTEMP2.
C
        IF (PLOC1.EQ.0.AND.PLOC2.EQ.0) GO TO 20
        IF (PLOC1.EQ.0.AND.PLOC2.NE.0) GO TO 30
        IF (PLOC1.NE.0.AND.PLOC2.EQ.0) GO TO 40
C
C Case of E(alfa).NE.E(beta) and both < 0.
C
        DO 50 I = 0, PLOC1-1
           HTEMP1(I) = HCODE1(I)
50      CONTINUE
        DO 60 I = PLOC1, R-1
           HTEMP1(I) = HCODE1(I+1)
60      CONTINUE
C
        DO 70 I = 0, PLOC2-1
           HTEMP2(I) = HCODE2(I)
70      CONTINUE
        DO 80 I = PLOC2, R-1
           HTEMP2(I) = HCODE2(I+1)
80      CONTINUE
```

```
C If E(alfa) < E(beta), then check and remove any leading 0's in HTEMP2
C
        IF (PLOC1.GT.PLOC2) CALL CHK0(NUM2,DEN2,P,R,HTEMP2,HTEMP2,KZERO)
C
C If E(beta) < E(alfa), then CHK0 on HTEMP1
C
        IF (PLOC2.GT.PLOC1) CALL CHK0(NUM1,DEN1,P,R,HTEMP1,HTEMP1,KZERO)
        GO TO 100
C
C Case of E(alfa) = E(beta) = 0.
C
20      DO 90 I = PLOC1, R-1
           HTEMP1(I) = HCODE1(I+1)
           HTEMP2(I) = HCODE2(I+1)
90      CONTINUE
        GO TO 100
C
C Case of E(alfa) = 0 and E(beta) < 0.
C
30      DO 110 I = PLOC1, R-1
           HTEMP1(I) = HCODE1(I+1)
110     CONTINUE
C
        DO 120 I = 0, PLOC2-1
           HTEMP2(I) = HCODE2(I)
120     CONTINUE
        DO 130 I = PLOC2, R-1
           HTEMP2(I) = HCODE2(I+1)
130     CONTINUE
C
C CHK0 on HTEMP1
C
        CALL CHK0(NUM1,DEN1,P,R,HTEMP1,HTEMP1,KZERO)
        GO TO 100
C
C Case of E(alfa) < 0 and E(beta) = 0.
C
40      DO 140 I = 0, PLOC1-1
           HTEMP1(I) = HCODE1(I)
140     CONTINUE
        DO 150 I = PLOC1, R-1
           HTEMP1(I) = HCODE1(I+1)
150     CONTINUE
C
        DO 160 I = PLOC2, R-1
           HTEMP2(I) = HCODE2(I+1)
160     CONTINUE
C
C CHK0 on HTEMP2
C
        CALL CHK0(NUM2,DEN2,P,R,HTEMP2,HTEMP2,KZERO)
C
C Multiplication process
C
100     DO 170 I = 0, R-1
C
C          form partial products for each I.
```

```
            DO 180 J = 0, R-1
               HCODEM(J) = (HTEMP2(I)*HTEMP1(J)) + CM
               CM = CARRY(HCODEM(J),P)
               HCODEM(J) = MOD(HCODEM(J),P)
180         CONTINUE
C
C           shift right each partial product by I locations
C
            CALL RSHIFT(HCODEM,HSHIFT,I,R)
C
C           final product is the sum of previous (shifted) partial products
C
            DO 190 K = 0, R-1
               HTEMPM(K) = HTEMPM(K) + HSHIFT(K) + CA
               CA = CARRY(HTEMPM(K),P)
               HTEMPM(K) = MOD(HTEMPM(K),P)
190         CONTINUE
C
C           re-initialize carries
C
            CM = 0
            CA = 0
C
170      CONTINUE
C
C E(gama) = E(alfa) + E(beta) + k + 1
C
         PLOC = PLOC1 + PLOC2 - KZERO
C
C Rewriting the final product with the resulting p-adic point in its position
C
         HCODEM(PLOC) = -1
         IF (PLOC.EQ.0) GO TO 200
         DO 210 I = 0, PLOC-1
            HCODEM(I) = HTEMPM(I)
210      CONTINUE
200      DO 220 I = PLOC+1, R
            HCODEM(I) = HTEMPM(I-1)
220      CONTINUE
C
         RETURN
         END
C
C ---------- O ----------
C
         SUBROUTINE CHKO(NUM,DEN,P,R,PTEMP1,PTEMP2,ZCOUNT)
C
C Subroutine to check if any leading 0's exist in the code PTEMP1 ( which is
C devoid of its p-adic point) and, if so, it removes them by recalculating
C the code over an extended field of length REXT = R + ZCOUNT (+ 1 in the case
C of ZCOUNT odd) and then shifting the code left by ZCOUNT positions.
C       PLOC is the position of the p-adic point in the newly
C            calculated code over REXT
C
         INTEGER PTEMP1(0:20),PTEMP2(0:20),PCODE(0:20)
         INTEGER NUM,DEN,P,R,ZCOUNT,PLOC,REXT
C
         ZCOUNT = 0
```

```
C Counting the number of leading 0's
C
        DO 10 I = 1, R-1
           IF ((PTEMP1(I-1).EQ.0.AND.PTEMP1(I).NE.0)
     +     .OR.(PTEMP1(I-1).EQ.0.AND.PTEMP1(I).EQ.0)) GO TO 20
           GO TO 30
20         ZCOUNT = ZCOUNT + 1
10      CONTINUE
C
C Length of extended field. It must be even.
C
30      REXT = R + ZCOUNT
        IF (MOD(REXT,2).NE.0) REXT = REXT + 1
C
C Calculation of the extended code.
C
        CALL HCODE(NUM,DEN,P,REXT,PCODE)
C
C Locate and remove the p-adic point from the extended code.
C
        DO 40 I = 0, REXT
           IF (PCODE(I).EQ.-1) PLOC = I
40      CONTINUE
        IF (PLOC.EQ.0) GO TO 50
        DO 60 I = 0, PLOC-1
           PTEMP1(I) = PCODE(I)
60      CONTINUE
50      DO 70 I = PLOC, REXT-1
           PTEMP1(I) = PCODE(I+1)
70      CONTINUE
C
C Remove leading 0's by left shifting PTEMP1. The final code, containing no
C leading 0's, is stored in PTEMP2.
C
90      IF (PTEMP1(0).NE.0) GO TO 80
        CALL LSHIFT(PTEMP1,PTEMP2,1,REXT)
        GO TO 90
C
80      RETURN
        END
```

```
C                     ****************************************
C                     *                                      *
C                     * Segmented p-adic DIVISION routine    *
C                     *         and related routines         *
C                     *                                      *
C                     ****************************************
C
C
      SUBROUTINE HDIV(NUM1,DEN1,NUM2,DEN2,HCODE1,HCODE2,P,R,HCODED)
C
C Subroutine performing the division operation in the segmented p-adic
C field Qp of length R. The quotient HCODE1 / HCODE2 is stored in HCODED.
C The subroutine operates on codes devoid of the p-adic point and it
C restitutes the p-adic point in the final result according to the
C respective values of E(alfa) and E(beta).
C       REXT   is the extended segmented p-adic field
C       PLOC1  . .   loaction of the p-adic point in HCODE1
C       PLOC2  . .  .       . . .     .     . HCODE2
C       PLOC   . .  .       . . .     .     . the final quotient HCODED
C       KZER01 .  a zero counter for leading 0's in the dividend
C       KZER02 . . .   .      .  .      .  . .   divisor
C       FLAG0  . . flag pointing to the existence of any leading 0's in the
C              divisor, in order to check on the existence of any leading 0's
C              in the dividend
C       QTEMP  is the result of solving HCODE2(0)*QTEMP = HCODE1(0) MOD(P).
C              Its value is the result of function DSOLVE.
C       ZCOUNT is an intermediate zero counter in successive remainders. It is
C              only needed to adjust the complementation routine in case the
C              remainder is all 0's. (The complement is then 00...0 and not
C              544...4)
C       CA     is the carry arising from addition
C       CM     . .  .    .       .    multiplication
C HTEMP1 and HSHFT1 are equivalent in what regards the left-shift routine, and
C so are HTEMP2 and HSHFT2.
C
      INTEGER HCODE1(0:20),HCODE2(0:20),HTEMP1(0:20),HTEMP2(0:20),
     +        HSHFT1(0:20),HSHFT2(0:20),HTEMP (0:20),HCODED(0:20)
      INTEGER NUM1,DEN1,NUM2,DEN2,P,R,PLOC1,PLOC2,PLOC,KZER01,KZER02,
     +        FLAG0,QTEMP,ZCOUNT,CA,CM,CARRY,DSOLVE,REXT
      EQUIVALENCE (HTEMP1,HSHFT1)
      EQUIVALENCE (HTEMP2,HSHFT2)
C
C Initialization.
C
      KZER01 = 0
      KZER02 = 0
      FLAG0  = 0
      ZCOUNT = 0
      CM     = 0
      CA     = 0
C
C Locate relative positions of p-adic points
C
      DO 10 I = 0, R
         IF (HCODE1(I).EQ.-1) PLOC1 = I
         IF (HCODE2(I).EQ.-1) PLOC2 = I
10    CONTINUE
```

```
C Rewrite HCODE1 and HCODE2 without their p-adic point in HTEMP1 and HTEMP2
C
        IF (PLOC1.EQ.0) GO TO 20
        DO 30 I = 0, PLOC1-1
           HTEMP1(I) = HCODE1(I)
30      CONTINUE
C
20      DO 40 I = PLOC1, R-1
           HTEMP1(I) = HCODE1(I+1)
40      CONTINUE
C
        IF (PLOC2.EQ.0) GO TO 50
        DO 60 I = 0, PLOC2-1
           HTEMP2(I) = HCODE2(I)
60      CONTINUE
C
50      DO 70 I = PLOC2, R-1
           HTEMP2(I) = HCODE2(I+1)
70      CONTINUE
C
C Check on leading 0's in the divisor. Accordingly, KZERO2 is incremented
C and the leading 0's flag is set.
C
        DO 80 I = 1, R-1
           IF ((HTEMP2(I-1).EQ.0.AND.HTEMP2(I).NE.0)
     +        .OR.(HTEMP2(I-1).EQ.0.AND.HTEMP2(I).EQ.0)) GO TO 90
           GO TO 100
90         KZERO2 = KZERO2 + 1
           FLAG0  = 1
80      CONTINUE
C
C If any leading 0's are found, an extended HCODE2 over REXT is computed.
C REXT has to be even.
C
100     IF (KZERO2.EQ.0) GO TO 110
        REXT = R + KZERO2
        IF (MOD(REXT,2).NE.0) REXT = REXT + 1
C
        CALL HCODE(NUM2,DEN2,P,REXT,HCODE2)
C
C Take p-adic point out of HCODE2
C
        IF (PLOC2.EQ.0) GO TO 120
        DO 130 I = 0, PLOC2-1
           HTEMP2(I) = HCODE2(I)
130     CONTINUE
120     DO 140 I = PLOC2, REXT-1
           HTEMP2(I) = HCODE2(I+1)
140     CONTINUE
C
C Shift HTEMP2 left by the number of leading 0's. The shifted sequence
C is called HSHFT2.
C
160     IF (HTEMP2(0).NE.0) GO TO 150
        CALL LSHIFT(HTEMP2,HSHFT2,1,REXT)
        GO TO 160
```

```
150      DO 170 I = 0, R-1
            HTEMP2(I) = HSHFT2(I)
170      CONTINUE
C
C If no leading 0's are found in the divisor, then proceed with the division
C directly. Otherwise, test for leading 0's in the dividend and increment
C KZERO1 accordingly.
C
         IF (FLAG0.EQ.0) GO TO 110
         DO 180 I = 1, R-1
            IF ((HTEMP1(I-1).EQ.0.AND.HTEMP1(I).NE.0)
     +      .OR.(HTEMP1(I-1).EQ.0.AND.HTEMP1(I).EQ.0)) GO TO 190
            GO TO 200
190         KZERO1 = KZERO1 + 1
180      CONTINUE
C
C Compute the extended HCODE1 over the same REXT above.
C
200      CALL HCODE(NUM1,DEN1,P,REXT,HCODE1)
C
C Take p-adic point out of HCODE1
C
         IF (PLOC1.EQ.0) GO TO 210
         DO 220 I = 0, PLOC1-1
            HTEMP1(I) = HCODE1(I)
220      CONTINUE
210      DO 230 I = PLOC1, REXT-1
            HTEMP1(I) = HCODE1(I+1)
230      CONTINUE
C
C Shift HTEMP1 left by the number of leading 0's. The sifted sequence
C is called HSHFT1.
C
250      IF (HTEMP1(0).NE.0) GO TO 240
         CALL LSHIFT(HTEMP1,HSHFT1,1,REXT)
         GO TO 250
C
240      DO 260 I = 0, R-1
            HTEMP1(I) = HSHFT1(I)
260      CONTINUE
C
C Division process
C
110      DO 270 K = 0, R-1
C
C           solve beta(0)*q(0) = alfa(0) mod(p)
            QTEMP = DSOLVE(HTEMP1(0),HTEMP2(0),P)
            HCODED(K) = QTEMP
C
C           no need to go through the remaining stages if we have reached the
C           last quotient digit
C
            IF (K.EQ.R-1) GO TO 300
```

```
            DO 280 I = 0, R-1-K
C
C              multiply QTEMP by divisor
C
               HTEMP(I) = (QTEMP * HTEMP2(I)) + CM
               CM = CARRY(HTEMP(I),P)
               HTEMP(I) = MOD(HTEMP(I),P)
C
C              keep count of number of 0's
C
               IF (HTEMP(I).EQ.0) ZCOUNT = ZCOUNT + 1
C
280         CONTINUE
C
C           if all R-K digits in the remainder are 0's, then the corresponding
C           complement is also 0 all over. Otherwise calculate complement.
C
            IF (ZCOUNT.NE.R-K) CALL HCOMP(HTEMP,P,R-K,HTEMP)
C
C           add complement to previous HTEMP1
C
            DO 290 I = 0, R-1-K
               HTEMP1(I) = HTEMP1(I) + HTEMP(I) + CA
               CA = CARRY(HTEMP1(I),P)
               HTEMP1(I) = MOD(HTEMP1(I),P)
290         CONTINUE
C
C           eliminate 1st. zero resulting from the 'subtraction' and
C           re-initialize variables.
C
            CALL LSHIFT(HTEMP1,HTEMP1,1,R-K)
            CM = 0
            CA = 0
            ZCOUNT = 0
C
270      CONTINUE
C
C Duplicate HCODED in HTEMP
C
300      DO 310 I = 0, R-1
            HTEMP(I) = HCODED(I)
310      CONTINUE
C
C Position of final p-adic point
C
         PLOC = PLOC1 - PLOC2
C
C If E(alfa) of dividend > E(beta) of divisor, then m(gama) is shifted right
C by a number of locations equal to the difference between E(alfa) and E(beta).
C In this case, E(gama) = E(alfa). Otherwise, E(gama) = E(alfa) - E(beta).
C Also, the number of leading 0's in the divisor contributes to the final
C positioning of the p-adic point in the quotient.
C
         IF (PLOC.LT.0) GO TO 320
         PLOC = PLOC + KZERO2 - KZERO1
         GO TO 330
```

```
320      K = IABS(PLOC)
         CALL RSHIFT(HCODED,HTEMP,K,R)
         PLOC = PLOC1 + KZERO2 - KZERO1
C
C Rewrite the final quotient with the resulting p-adic point in its position
C
330      HCODED(PLOC) = -1
         IF (PLOC.EQ.0) GO TO 340
         DO 350 I = 0, PLOC-1
            HCODED(I) = HTEMP(I)
350      CONTINUE
340      DO 360 I = PLOC+1, R
            HCODED(I) = HTEMP(I-1)
360      CONTINUE
C
         RETURN
         END
C
C ---------- O ----------
C
         INTEGER FUNCTION DSOLVE(A,B,P)
C
C Function returning the solution of
C        B * DSOLVE = A (mod P)
C given A (here, corresponding to 1st. digit in Hensel code of dividend),
C B (corresponding to 1st. digit in Hensel code of divisor) and P. The result
C is one of the elements in the quotient of alfa / beta.
C
         INTEGER A,B,P,SOL
C
         DO 10 I = 0, P-1
            SOL = MOD((B * I),P)
            IF (SOL.NE.A) GO TO 10
            DSOLVE = I
            GO TO 20
10       CONTINUE
C
20       RETURN
         END
```

```
C               ************************************************
C               *                                              *
C               * HENSEL CODE to RATIONAL CONVERSION routine   *
C               *            and related routines              *
C               *                                              *
C               ************************************************
C
C
      SUBROUTINE CONVRT(HCODE,P,R,NUM,DEN)
C
C Subroutine to convert a given Hensel code, HCODE(P,R), into its rational
C equivalent by a successive addition process.
C       PLOC   is the location of the p-adic point
C       ACOUNT is a counter for the number of successive additions
C              performed
C       NEGTV  is a flag to indicate that the last R/2 digits = P-1 and
C              hence the rational is negative
C
      INTEGER HCODE(0:20),HTEMP(0:20)
      INTEGER P,R,NUM,DEN,PLOC,C,CARRY,ACOUNT,NEGTV
C
C Initialization
C
      ACOUNT = 0
      NUM    = 0
      DEN    = 0
      NEGTV  = 0
C
C Location of the p-adic point
C
      DO 10 I = 0, R
         IF (HCODE(I).EQ.-1) PLOC=I
10    CONTINUE
C
C Rewriting the Hensel code in HTEMP without the p-adic point
C
      IF (PLOC.EQ.0) GO TO 20
      DO 30 I = 0, PLOC-1
         HTEMP(I) = HCODE(I)
30    CONTINUE
C
20    DO 40 I = PLOC, R-1
         HTEMP(I) = HCODE(I+1)
40    CONTINUE
C
C Duplicating HTEMP by putting it in HCODE for the successive additions
C
      DO 50 I = 0, R-1
         HCODE(I) = HTEMP(I)
50    CONTINUE
C
C Check whether last R/2 digits = 0μ in which case the successive additions
C process is ended and the rational is +ve.
C
120   DO 60 I = R/2, R-1
         IF (HTEMP(I).NE.0)     GO TO 70
60    CONTINUE
```

```
        GO TO 100
C
C Check whether last R/2 digits = P-1μ in which case the successive additions
C process is ended and NEGTV is set to 1 to indicate that the rational is -ve.
C
70      DO 80 I = R/2, R-1
           IF (HTEMP(I).NE.(P-1)) GO TO 90
80      CONTINUE
        NEGTV = 1
        GO TO 100
C
C Segmented p-adic addition similar to HADD
C
90      C = 0
        DO 110 I = 0, R-1
           HTEMP(I) = HCODE(I) + HTEMP(I) + C
           C = CARRY(HTEMP(I),P)
           HTEMP(I) = MOD(HTEMP(I),P)
110     CONTINUE
C
C Addition counter is increased by 1 for each performed addition
C
        ACOUNT = ACOUNT + 1
        GO TO 120
C
C Final denominator is: (number of additions + 1) x (P ** PLOC)
C
100     DEN = (ACOUNT + 1) * (P ** PLOC)
C
C If last R/2 digits = 0 (indicated by NEGTV = 0), then the final
C numerator is the +ve weighted sum of the leading R/2 digits.
C Otherwise, if last R/2 digits = P-1 (i.e., NEGTV = 1), then the
C final numerator is the -ve weighted sum of the leading R/2 digits
C in the complement of HTEMP.
C
        IF (NEGTV.EQ.0) GO TO 130
        CALL HCOMP(HTEMP,P,R,HTEMP)
130     DO 140 I = 0, (R/2)-1
           NUM = NUM + HTEMP(I) * P**I
140     CONTINUE
        IF (NEGTV.EQ.1) NUM = -NUM
C
        RETURN
        END
```