Since $I_0 = P_0Q_{-1} - Q_0P_{-1} = 1 \cdot 1 - 0 \cdot 0 = 1$, one has $I_k = (-1)^k$. It follows that

$$S_k - S_{k-1} = \frac{P_k}{Q_k} - \frac{P_{k-1}}{Q_{k-1}} = \frac{(-1)^k}{Q_kQ_{k-1}}, \qquad k > 1$$

or

$$P_kQ_{k-1} - Q_kP_{k-1} = (-1)^k, \qquad k > 1. \tag{10A}$$

If GCD $(P_k,Q_k) = d_k$, then, by (10A), $d_k \mid (-1)^k$. This implies that $d_k = 1$. Hence, GCD $(P_k,Q_k) = 1$.

A simple example showing how to compute the rational approximations to an irreducible rational number is presented in tabular form in Table II. For this example, S is the fraction 38/105. From the tabular form, when $k = n = 6$, one observes $R_6 = 0$. By (7A), $S = S_6 = P_6/Q_6 = 38/105$. For a more detailed discussion of the relation of Euclid's algorithm to the continued fraction associated with a rational element in the field of real numbers, see [12].

REFERENCES

[1] W. C. Gore, "Transmitting binary symbols with Reed–Solomon code," Johns Hopkins Univ. EE Report No. 73-5, Baltimore, MD, Apr. 1973.

[2] A. Michelson, "A new decoder for the Reed–Solomon codes using a fast transform technique," Systems Engineering Technical Memorandum No. 52, Electronic Systems Group, Eastern Division GTE Sylvania, Waltham, MA, Aug. 1975.

[3] D. Mandelbaum, "On decoding Reed–Solomon codes," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 707–712, Nov. 1971.

[4] W. W. Peterson, *Error-Correcting Codes.* Cambridge, MA: M.I.T. Press, 1961, pp. 168–169.

[5] C. M. Rader, "Discrete convolution via mersenne transforms," *IEEE Trans. Comput.*, vol. C-21, pp. 1269–1273, Dec. 1972.

[6] R. C. Agarwal and C. S. Burrus, "Number theoretic transform to implement fast digital convolution," in *Proc. IEEE*, vol. 63, pp. 550–560, Apr. 1975.

[7] I. S. Reed and T. K. Truong, "Convolutions over residue classes of quadratic integers," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 468–475, July 1976.

[8] J. H. MacClellan, "Hardware realization of a Fermat number transform," *IEEE Trans. on Acoustics Speech, and Signal Processing*, vol. ASSP-24, pp. 216–225, June 1976.

[9] J. Justesen, "On the complexity of decoding of Reed–Solomon codes," *IEEE Trans. Inform Theory*, vol. IT-22, pp. 237–238, Mar. 1976.

[10] I. S. Reed, T. K. Truong, and L. R. Welch, "The fast decoding of Reed–Solomon codes using number theoretic transforms," in the Deep Space Network Progress Report 42-35, Jet Propulsion Laboratory, Pasadena, CA, July 1976, pp. 64–78.

[11] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw-Hill, 1968, Ch. 7.

[12] I. M. Vinogrodov, *Elements of Number Theory.* New York: Dover, 1954, Ch. 1.

Correspondence

An Improved Algorithm for Computing Logarithms over $GF(p)$ and Its Cryptographic Significance

STEPHEN C. POHLIG AND MARTIN E. HELLMAN, MEMBER, IEEE

***Abstract*—A cryptographic system is described which is secure if and only if computing logarithms over $GF(p)$ is infeasible. Previously published algorithms for computing this function require $O(p^{1/2})$ complexity in both time and space. An improved algorithm is derived which requires $O(\log^2 p)$ complexity if $p - 1$ has only small prime factors. Such values of p must be avoided in the cryptosystem. Constructive uses for the new algorithm are also described.**

I. INTRODUCTION

This note considers the pair of inverse functions

$$y \equiv \alpha^x \pmod p \tag{1}$$

$$x \equiv \log_\alpha y \text{ over } GF(p) \tag{2}$$

which are referred to as the exponential and logarithmic functions to the base α, modulo p, where p is prime, and α is a fixed primitive element of $GF(p)$. Since α is primitive, x and y are in a one-to-one correspondence for integer values in the range $1 \le x,y \le p - 1$.

It is well-known [1, p. 399] that exponentiation mod p is computable with at most $2\lceil \log_2 p \rceil$ multiplications mod p, and with only three words of memory, each $\lceil \log_2 p \rceil$ bits long, where $\lceil \cdot \rceil$ denotes the smallest integer equal to or greater than the enclosed number. (All logarithms not expressly mod p are over the reals and are to the base 2.) To give the flavor of the algorithm, note that

$$\alpha^{18} = (((\alpha^2)^2)^2)^2 \cdot \alpha^2. \tag{3}$$

The inverse problem of computing logarithms mod p is believed to be much harder, and the best previously published algorithm [2, p. 9] requires $2\lceil p^{1/2} \rceil$ multiplications mod p, in addition to other operations of comparable complexity. This algorithm also requires $2\lceil \sqrt{p} \rceil$ words of memory, each $\lceil \log_2 p \rceil$ bits long.

Exponentiation mod p might thus be a *one-way function.* An invertible function f is said to be one-way if it is easy to compute $y = f(x)$ for all x in the domain, but it is computationally infeasible to compute $f^{-1}(y)$ for almost all y in the range of f.

We have deliberately not given a precise definition of a computation being "easy" or "infeasible." In 1950, a computation requiring one million instructions and 10 000 words of memory could not have been called "easy," while today it can be accomplished in a few seconds on a small computer. Similarly, a computation which requires 10^{30} operations is infeasible today, but will probably not even be difficult a hundred years hence. A precise definition of a one-way function would therefore vary with time and technology. It may be possible to avoid this problem by using a currently acceptable definition of easy and a physics-

Manuscript received June 17, 1976; revised April 14, 1977. This work was supported in part by the National Science Foundation under Grant ENG 10173, and in part by the Fannie and John Hertz Foundation.

S. C. Pohlig was with the Department of Electrical Engineering, Stanford University, Stanford, CA. He is now with the M.I.T. Lincoln Laboratory, Lexington, MA 02173.

M. E. Hellman is with the Department of Electrical Engineering, Stanford University, Stanford, CA 94305.

0018-9448/78/0100-0106$00.75 © 1978 IEEE

limited definition of infeasible. Any computation that is easy today will be no harder in the future, and a 10^{60} bit memory will always be unattainable because its construction requires more mass than exists in the solar system, even if only one molecule is needed per bit of memory. Thermodynamics places a limit of approximately 10^{70} on the number of operations which can be performed even if the entire energy output of the sun could be harnessed forever [3], [4]. We prefer to avoid such conservative definitions, however, because they may exclude practically valuable one-way functions. It will be seen, however, that exponentiation mod p may be able to satisfy even the most conservative definition of a one-way function.

Currently, the primary use for one-way functions is in protecting the password file in a time-shared computer system [5]–[7]. They have other related uses [8], [9]. Their existence is also necessary to the existence of secure cryptographic systems, because any secure cryptosystem can be used to produce a one-way function [9]; while the converse is not true in general, Section II of this paper describes a cryptographic system which is secure if and only if exponentiation mod p is one-way.

Sections III and IV develop an improved algorithm for computing logarithms over $GF(p)$. This algorithm has complexity not much greater than that required for exponentiation mod p, when $p - 1$ has only small prime factors, but is infeasible to compute when $p - 1$ has a large prime factor. Although not previously published, the new algorithm was discovered independently by Roland Silver some years ago, and more recently by Richard Schroeppel and H. Block.

The improved algorithm dictates that $p - 1$ must have a large prime factor if exponentiation mod p is to be used as a one-way function or in a cryptosystem. Of course, just because Knuth's algorithm and the new one are not computable in practice for certain values of p does not mean that there are not more efficient algorithms for these values of p that are as yet undiscovered.

A second use of the improved algorithm is in problems where it would be useful to rapidly compute logarithms over $GF(p)$ for arbitrary but large values of p. By choosing p to obtain the full benefit of the improved algorithm, very large primes can be used. Scholtz and Welch [10] devised a multiple access code which required the computation of logarithms over $GF(p)$, and Merkle and Hellman [11] have devised a public-key distribution system which utilizes our algorithm for computing logarithms over $GF(p)$.

Section V gives examples of primes at both extremes. A 137 digit prime is given for which the new algorithm is easily implemented, and a 60 digit prime is given for which no known algorithm can be implemented.

II. Use in Cryptography

It is well-known [12, p. 37] that if p is prime then

$$z^{p-1} \equiv 1 \pmod{p}, \qquad 1 \le z \le p - 1. \tag{4}$$

Consequently, arithmetic in the exponent is done modulo $p - 1$, *not* modulo p. That is,

$$z^x \equiv z^{x(\bmod p-1)} \pmod{p} \tag{5}$$

for all integers x.

Based on this observation, we can construct a cryptosystem. Let M, K, and C denote the plaintext message, key, and ciphertext (cryptogram) with the restrictions

$$1 \le M \le p - 1, \tag{6}$$

$$1 \le C \le p - 1, \tag{7}$$

$$1 \le K \le p - 2, \tag{8}$$

$$\mathrm{GCD}\,(K, p - 1) = 1. \tag{9}$$

In practice, M would probably be limited to be an l bit integer where $l = \lfloor \log_2 (p - 1) \rfloor$. Also, restrictions might be imposed on K (e.g., $K \neq 1$) to avoid simple but improbable transformations. Condition (9) guarantees that

$$D \equiv K^{-1} \pmod{p - 1} \tag{10}$$

is well-defined with

$$1 \le D \le p - 2. \tag{11}$$

Now let

$$C \equiv M^K \pmod{p} \tag{12}$$

be the enciphering operation. Then

$$M \equiv C^D \pmod{p} \tag{13}$$

is the deciphering operation. Both operations are easily computed and involve only one exponentiation mod p (equivalently $2\lceil \log_2 p \rceil$ multiplications mod p). Computing D from K need be done only once and also requires on the order of $\log_2 p$ operations using Euclid's algorithm [1, Section 4.5.2].

Finding the key through cryptanalysis, on the other hand, is equivalent to computing a logarithm over $\mathrm{GF}(p)$ and is thus impossible if and only if exponentiation mod p is a one-way function. This is because

$$K \equiv \log_M C \text{ over } \mathrm{GF}(p) \tag{14}$$

so that, even if the cryptanalyst has the advantage of knowing a plaintext-ciphertext pair, it is as hard to find the key as to find a logarithm over $GF(p)$. Such a known plaintext cryptanalytic attack [9] is a standard test applied to certify a system as secure. It and variations of it occur in practice as well.

Note that M must be a primitive element of $GF(p)$ for M and C to uniquely determine K. We now show that if M is not primitive, or if the cryptanalyst has a number of randomly chosen $M - C$ pairs all related by the same key, then his task is not lightened. To see this, observe that if M and C are related by K, that is,

$$C \equiv M^K \pmod{p}, \tag{15}$$

then so are M' and C', where

$$M' \equiv M^n \pmod{p} \tag{16}$$

and

$$C' \equiv C^n \pmod{p} \tag{17}$$

for any integer n. Possession of a single $M - C$ pair with M primitive thus allows the cryptanalyst easily to generate a large number of $M' - C'$ pairs also related by K. Building a table is precluded by using values of p on the order of 2^{100} or larger.

We have not been able to show that the above system can resist a chosen plaintext attack [9] in which the cryptanalyst gets to choose M and see the corresponding C, but neither have we found a way for him to use this option to advantage over the known plaintext attack.

There is a minor problem with the above cryptosystem in that the most natural representation of K is as an integer between 1 and $p - 1$, but not all such integers are allowable keys because of the restriction that $\mathrm{GCD}\,(K, p - 1) = 1$. Fortunately, the fraction ρ of usable keys is not too small. From the definition of $\phi(n)$, the Euler totient function,

$$\rho = \phi(p - 1)/(p - 1), \tag{18}$$

where $\phi(n)$ is the number of positive integers not exceeding and relatively prime to n. It is known [13, p. 826] that

$$\rho = \prod_{p_i \mid (p-1)} (1 - p_i^{-1}), \tag{19}$$

where $\{p_i\}$ are the prime divisors of $p - 1$. As noted in Section V, primes of the form $p = 2p' + 1$, with p' prime, are the most promising candidates for yielding a secure cryptosystem. Then, with p large,

$$\rho = (1/2)[1 - (1/p')] \approx 1/2. \tag{20}$$

Even if $p \neq 2p' + 1$, a reasonable fraction of keys is usable because, for all $p < 1.6 \times 10^{103}$, ρ is greater than 0.1. For this range of p, at most ten tries are needed on the average to find a suitable key. The test involves Euclid's algorithm and is therefore no more complex than computing D from K. The effective loss in key size is less than 4 bits, which is also quite negligible. The real question concerning the utility of this cryptosystem is the difficulty of computing logarithms over $GF(p)$.

Rivest, Shamir, and Adleman [14] have noted that it can be advantageous to perform the enciphering and deciphering operations (12) and (13) modulo n, where n is the product of two large primes. Then, instead of (10), one uses $D = K^{-1} \pmod{\phi(n)}$. By keeping the factorization of n secret, it is possible to generate a public key cryptosystem as defined in [9]. The enciphering key (K,n) can be made public without compromising the deciphering key (D,n), and yet it is easy for anyone to generate a pair of enciphering and deciphering keys. Public key cryptosystems eliminate the key distribution problem and allow the generation of true digital signatures [9]. See [14] for details.

Another application of exponentiation modulo p to cryptography is given in [9]. There it is shown how these functions can be used to securely "transmit" a key over an insecure channel with no previous exchange of secret information. This capability is a direct result of the commutivity of enciphering under two different keys K_1 and K_2, i.e.,

$$(M^{K_1})^{K_2} = (M^{K_2})^{K_1}. \tag{20}$$

The reader is referred to [9] for details.

III. An Algorithm for $p = 2^n + 1$

The improved algorithm for computing logarithms mod p is best understood by first considering the special case $p = 2^n + 1$. We are given α, p, and y, with α a primitive element of $GF(p)$, and must find x such that $y \equiv \alpha^x \pmod{p}$. We can assume $0 \leq x \leq p - 2$, since $x = p - 1$ is indistinguishable from $x = 0$.

When $p = 2^n + 1$, x is easily determined by finding the binary expansion $\{b_0, \cdots, b_{n-1}\}$ of x, i.e.,

$$x = \sum_{i=0}^{n-1} b_i 2^i. \tag{21}$$

The least significant bit b_0 of x is determined by raising y to the $(p - 1)/2 = 2^{n-1}$ power and applying the rule

$$y^{(p-1)/2} \pmod{p} \equiv \begin{cases} +1, & b_0 = 0 \\ -1, & b_0 = 1. \end{cases} \tag{22}$$

This fact is established by noting that, since α is primitive,

$$\alpha^{(p-1)/2} \equiv -1 \pmod{p}, \tag{23}$$

and therefore

$$y^{(p-1)/2} \equiv (\alpha^x)^{(p-1)/2} \equiv (-1)^x \pmod{p}. \tag{24}$$

The next bit in the expansion of x is then determined by letting

$$z \equiv y\alpha^{-b_0} \equiv \alpha^{x_1} \pmod{p}, \tag{25}$$

where

$$x_1 = \sum_{i=1}^{n-1} b_i 2^i. \tag{26}$$

Clearly, x_1 is a multiple of four if and only if $b_1 = 0$. If $b_1 = 1$, then x_1 is divisible by two but not by four. Reasoning as before, we obtain

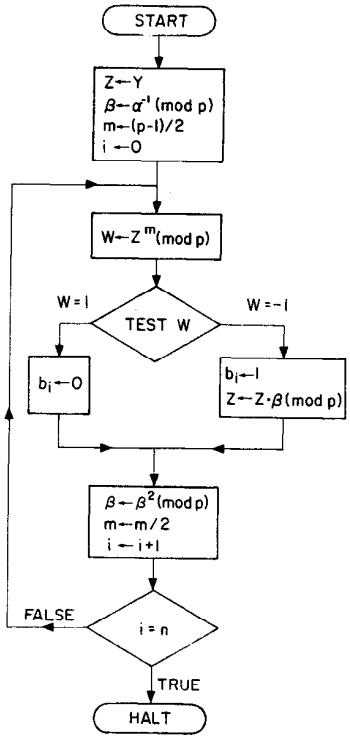

Fig. 1. Flowchart for algorithm when $p = 2^n + 1$.

$$z^{(p-1)/4} \pmod{p} \equiv \begin{cases} +1, & b_1 = 0 \\ -1, & b_1 = 1. \end{cases} \tag{27}$$

The remaining bits of x are determined in a similar manner. This algorithm is summarized in the flowchart of Fig. 1. To aid in understanding this flowchart, note that, at the start of the ith loop,

$$m = (p - 1)/2^{i+1} \tag{28}$$

and

$$z \equiv \alpha^{x_i} \pmod{p}, \tag{29}$$

where

$$x_i = \sum_{j=i}^{n-1} b_j 2^j. \tag{30}$$

Thus raising z to the mth power gives

$$\begin{aligned} z^m &\equiv \alpha^{(x_i m)} \equiv \alpha^{[(p-1)/2]\cdot(x_i/2^i)} \\ &\equiv (-1)^{x_i/2^i} \equiv (-1)^{b_i} \pmod{p}, \end{aligned} \tag{31}$$

so that $z^m \equiv 1 \pmod{p}$ if and only if $b_i = 0$, and $z^m \equiv -1 \pmod{p}$ if and only if $b_i = 1$.

IV. An Algorithm for Arbitrary Primes

We now generalize the above algorithm to arbitrary primes p; this generalization is necessary because $2^{16} + 1$ is the largest known prime of the form $2^n + 1$. Let

$$p - 1 = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}, \qquad p_i < p_{i+1} \tag{32}$$

be the prime factorization of $p - 1$, where the p_i are distinct primes and the n_i are positive integers. The value of $x \pmod{p_i^{n_i}}$ will be determined for $i = 1, \cdots, k$, and the results will then be combined via the Chinese remainder theorem [12, p. 48] to ob-

tain

$$x \left(\text{mod} \prod_{i=1}^{k} p_i^{n_i}\right) = x \pmod{p-1} = x \tag{33}$$

since $0 \le x \le p-2$. The Chinese remainder theorem can be implemented in $O(k \log_2 p)$ operations and $O(k \log_2 p)$ bits of memory. (We count a multiplication mod p as one operation.)

Consider the following expansion of $x \pmod{p_i^{n_i}}$:

$$x \pmod{p_i^{n_i}} = \sum_{j=0}^{n_i-1} b_j p_i^j, \tag{34}$$

where $0 \le b_j \le p_i - 1$. The least significant coefficient, b_0, is determined by raising y to the $(p-1)/p_i$ power,

$$y^{(p-1)/p_i} \equiv \alpha^{(p-1)x/p_i} \equiv \gamma_i^x \equiv (\gamma_i)^{b_0} \pmod{p} \tag{35}$$

where

$$\gamma_i = \alpha^{(p-1)/p_i} \tag{36}$$

is a primitive p_ith root of unity. There are therefore only p_i possible values for $y^{(p-1)/p_i} \pmod{p}$, and the resultant value uniquely determines b_0.

The next digit b_1 in the base p_i expansion of $x \pmod{p_i^{n_i}}$ is determined in a manner similar to that of Section III. Let

$$z \equiv y \cdot \alpha^{-b_0} \equiv \alpha^{x_1} \pmod{p}, \tag{37}$$

where

$$x_1 = \sum_{j=1}^{n_i-1} b_j p_i^j. \tag{38}$$

Now, raising z to the $(p-1)/p_i^2$ power yields

$$z^{(p-1)/p_i^2} \equiv \alpha^{(p-1)\cdot x_i/p_i^2} \equiv \gamma_i^{x_1/p_i} \equiv (\gamma_i)^{b_1} \pmod{p}. \tag{39}$$

Again, there are only p_i possible values of $z^{(p-1)/p_i^2}$, and this value determines b_1. This process is continued to determine all the coefficients b_j.

The flowchart of Fig. 2 summarizes the algorithm for computing the coefficients $\{b_j\}$ of the expansion (34). This algorithm is used k times to compute $x \pmod{p_i^{n_i}}$ for $i = 1,2,\cdots,k$, and these results are combined by the Chinese remainder theorem to obtain x. The function $g_i(w)$ in Fig. 2 is defined by

$$\gamma_i^{g_i(w)} \equiv w \pmod{p}, \qquad 0 \le g_i(w) \le p_i - 1, \tag{40}$$

where γ_i is defined in (36).

If all the prime factors $\{p_i\}_{i=1}^k$ of $p-1$ are small, then the $g_i(w)$ functions are easily implemented as tables, and computing a logarithm over $GF(p)$ requires $O(\log_2 p)^2$ operations and only minimal memory for the $g_i(w)$ tables. The dominant computational requirement is computing $w = z^n$, which requires $O(\log_2 p)$ operations. This loop is traversed $\Sigma_{i=1}^k n_i$ times, and, if all p_i are small, $\Sigma_{i=1}^k n_i$ is approximately $\log_2 p$—it is never larger than this. Thus when $p-1$ has only small prime factors, exponentiation mod p is not a good one-way function.

If, however, $p-1$ possesses a large prime factor p_k, then computation of $g_i(w)$ when $i = k$ is the dominant computational requirement of this algorithm. The following lemma treats this complexity, dropping the subscript i for notational convenience.

Lemma: Let

$$w \equiv \gamma^g \pmod{p} \tag{41}$$

with $0 \le g \le p_i - 1$, with $\gamma = \alpha^{(p-1)/p_i}$, and with α primitive. Then, for given α and w, g can be computed with $O(p_i^{(1-r)}(1 + \log_2 p_i^r))$ operations which depend on w, and with $O(p_i^r \log_2 p)$ bits of memory, for any $0 \le r \le 1$. Unless $r > 1/2$, precomputation requiring $O(p_i^r \log_2 p_i^r)$ operations is unimportant when compared to the number of w-dependent operations. A multiplication or addition mod p is counted as a single operation.

Proof: The proof uses a generalization of Knuth's algorithm for computing logarithms over $GF(p)$ in $O(p^{1/2} \log_2 p)$ operations

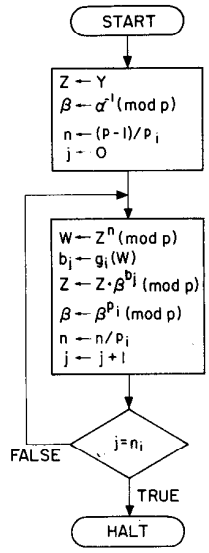

Fig. 2. Flowchart for algorithm when $p = \Pi_i p_i^{n} + 1$.

and bits of memory [2, p. 9]. Let

$$m = \lceil p_i^r \rceil. \tag{42}$$

Then there exist integers c and d such that

$$g = cm + d, \tag{43}$$

with

$$0 \le c < \lceil p_i/m \rceil \approx p_i^{1-r} \tag{44}$$

and

$$0 \le d < m \approx p_i^r. \tag{45}$$

Solving (41) for g is equivalent to finding c and d such that

$$\gamma^d \equiv w\gamma^{-cm} \pmod{p}. \tag{46}$$

To find c and d, we can therefore precompute $\gamma^d \pmod{p}$ for $d = 0,1,2,\cdots,m-1$ (in $O(p_i^r)$ operations), and then sort the resulting values (in $O(p_i^r \log_2 p_i^r)$ operations). Next we compute w, $w\gamma^{-m}, w\gamma^{-2m}, \cdots \pmod{p}$ and check for a match with the sorted table of $\{\gamma^d\}$. Each value of c tried requires one multiplication mod p and $\log_2 p_i^r$ comparisons, or $(1 + \log_2 p_i^r)$ operations all together. There are $O(p_i^{1-r})$ values of c to be tried. Q.E.D.

When $r = 1$, $g(w)$ is implemented as a table lookup. When $r = 0$, $g(w)$ is found by computing γ^g for $g = 0,1,2,\cdots,p_i - 1$ until $\gamma^g \equiv w \pmod{p}$. The lemma shows that, neglecting logarithmic factors, the time-memory product can be held constant (since $p_i^r p_i^{1-r} = p_i$) as we go between the extremes of a table lookup and an exhaustive trial-and-error search.

Since memory tends to be more costly than computation, values of $r < 1/2$ are of most interest, and then the precomputational effort is not important.

Theorem: Let

$$p - 1 = p_1^{n_1} p_2^{n_2} \cdots p_k^{n_k}, \qquad p_i < p_{i+1} \tag{47}$$

be the prime factorization of $p-1$, where p is prime, the p_i are distinct primes, and $n_i \ge 1$. Then, for any $\{r_i\}_{i=1}^k$ with all $0 \le r_i \le 1$, logarithms over $GF(p)$ can be computed in $O(\Sigma_{i=1}^k n_i[\log_2 p + p_i^{1-r_i}(1 + \log_2 p_i^{r_i})])$ operations with $O(\log_2 p \Sigma_{i=1}^k (1 + p_i^{r_i}))$ bits of memory. Precomputation requires $O(\Sigma_{i=1}^k (p_i^{r_i} \log_2 p_i^{r_i} + \log_2 p))$ operations and is unimportant when $r_k < 1/2$.

Remark: If p is such that all p_i are small, then the required computational effort is approximately $\log_2 p$ times as great as that required to evaluate one exponential. Because cryptanalytic

effort should be at least 10^9 times as great as enciphering and deciphering effort, the cryptosystem of Section II must avoid using such values of p. However, when p_k, the largest prime factor of p, is comparable in size to p (e.g., $p - 1 = 2p_k$), then the new algorithm is almost no better than previously known algorithms.

Proof: The proof follows from the lemma and the observation that the Chinese remainder theorem requires $O(k)$ operations, $O(k \log_2 p)$ bits of memory, and $O(k \log_2 p)$ precomputation operations. The term $\Sigma_{i=1}^{k} n_i \log_2 p$ in the number of operations accounts for $w \leftarrow z^n \pmod p$ and $\beta \leftarrow \beta^{p_i} \pmod p$ in the loop of Fig. 2. The loop is executed $\Sigma_{i=1}^{k} n_i$ times.

V. Discussion

The new algorithm is most efficient when $p - 1$ has only small prime factors. For example, $p = (5^2 \cdot 2^{448} + 1)$ is prime [15, p. 51] and requires $2 + 448 = 450$ iterations of the loop shown in Fig. 2. The dominant computational requirement is for the 450 exponentiations mod p involved in computing w. Computing a logarithm mod p is thus only 450 times as hard as computing an exponential mod p, and the latter function is not one-way for this choice of p. In consequence, the cryptosystem and method of key exchange described in Section II are not secure for this choice of p. By comparison, previous algorithms [2, p. 9] require over 10^{65} operations and bits of memory to compute one logarithm over $GF(p)$ for this particular p.

The new algorithm is least efficient when $p = 2p' + 1$, where p' is also prime. In this case, the dominant computational requirement is to compute $g(w)$ when $p_i = p_2 = p'$. Indeed, once $g(w)$ is known, $x \equiv \log_\alpha y$ over $GF(p)$ is easily found since either $x = g(w)$, or $x = g(w) + p'$. When $p = 2p' + 1$, the new algorithm is essentially the same as Knuth's, the exception being that r need not be $1/2$. A computer program was used to search for large primes of this form. One such prime was $p = 2p' + 1$, with

$$p' = (2^{121} \cdot 5^2 \cdot 7^2 \cdot 11^2 \cdot 13 \cdot 17 \cdot 19 \cdot 23 \cdot 29 \cdot 31 \cdot 37 \cdot 41 \cdot 43 \cdot 47 \cdot 53 \cdot 59) + 1. \quad (48)$$

Both p and p' require 60 decimal digits, i.e., are about 200 bits long. A pair p and p' of 25 digit primes of this form was also found, namely

$$p' = (2^{13} \cdot 5 \cdot 7 \cdot 11 \cdot 13 \cdots 47 \cdot 53 \cdot 59) + 1. \quad (49)$$

Note that three cannot be a factor of $(p' - 1)$, since otherwise $2p' + 1$ would be divisible by three and hence not prime. (One can prove that the first p and p' are primes by checking that six and two are primitive elements of $GF(p')$ and $GF(p)$, respectively. For the second pair, three and two are the corresponding primitive elements. In both cases, these are the smallest primitive elements.)

When $p = 2p' + 1$ with the p' of (48) and $r = 1/2$, computing $g(w)$ by the method used in the lemma requires more than 10^{30} operations and more than 10^{30} bits of memory. By choosing another value of r, one or the other of these requirements can be reduced, but both cannot be done simultaneously. Since both requirements are infeasible today (and will be for the next 50 years even if current cost trends continue) the new algorithm for computing logarithms mod p is infeasible for this value of p and any choice of r. By going to a 200 digit prime of the form $2p' + 1$, one could invoke the previously mentioned physical arguments to preclude use of the new algorithm forever. In contrast, exponentiation requires only about 400 multiplications mod p and three 200 bit words of memory for the 60 digit value of p, and only 1330 multiplications mod p and three 665 bit words of memory when p is 200 digits long.

The new algorithm directly extends to all finite fields $GF(p^m)$ and, in particular, is infeasible for computing logarithms over $GF(2^m)$ when $2^m - 1$ is a large Mersenne prime. Using this choice of field for the cryptosystem of Section II also has the advantage that all keys $1 \leq K \leq 2^m - 2$ are usable, and that the plaintext message contains an integral number of bits. In $GF(2^{521})$ [15, p. 50], enciphering and deciphering require at most 1042 multiplications (in the field), while cryptanalysis using the new algorithm requires at least $2^{521/2} = 2.6 \times 10^{78}$ times as much computation.

If the new algorithm is at all close to optimal, it follows that values of p exist for which exponentiation mod p would be a one-way function, even under the most restrictive of definitions. By implication, the cryptosystem of Section II would be secure under the most conservative of definitions. The real question is whether there are as yet undiscovered algorithms which are vastly more efficient when $p - 1$ has a large prime factor.

We encourage research oriented toward finding such an algorithm or toward establishing a large lower bound on the computational effort for finding logarithms mod p. Because the security of this cryptosystem is equivalent to such an easily stated problem, there is more than the usual hope that its security can be established through mathematical proof, if indeed it is secure. Success in this endeavor would have a revolutionary impact on cryptography.

References

[1] D. E. Knuth, *The Art of Computer Programming, Vol. II: Seminumerical Algorithms.* Reading, MA: Addison-Wesley, 1969.

[2] D. E. Knuth, *The Art of Computer Programming, Vol. III: Sorting and Searching.* Reading, MA: Addison-Wesley, 1973.

[3] R. W. Keyes, "Physical limits in digital electronics," in *Proc. IEEE,* vol. 63, pp. 740–767, May 1975.

[4] R. W. Landauer, "Irreversibility and heat generation in the computing process," *IBM J. Res. Develop.,* vol. 5, pp. 183–191, 1961.

[5] M. V. Wilkes, *Time-Sharing Computer Systems.* New York: American Elsevier, 1968.

[6] A. Evans, W. Kantrowitz, and E. Weiss, "A user authentication scheme not requiring secrecy in the computer," *Comm. ACM,* vol. 17, pp. 437–442, Aug. 1974.

[7] G. B. Purdy, "A high security log-in procedure," *Comm. of ACM,* vol. 17, pp. 442–445, Aug. 1974.

[8] R. Merkle, "Secure communication over an insecure channel," submitted to *Comm. of ACM.*

[9] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inform Theory,* vol. IT-22, pp. 644–654, Nov. 1976.

[10] R. A. Scholtz and L. R. Welch, "Generalized residue sequence," in *Proc. Int. Conf. Comm.,* Seattle, WA, June 1973.

[11] R. Merkle and M. E. Hellman, "Hiding information and receipts in trap door knapsacks," presented at 1977 IEEE Int. Symp. Information Theory, Ithaca, NY, Oct. 1977; also to appear in *IEEE Trans. Inform. Theory.*

[12] R. G. Archibald, *An Introduction to the Theory of Numbers.* Columbus, OH: Merrill, 1970.

[13] *Handbook of Mathematical Functions,* M. Abramowitz and I. A. Stegun, Eds. New York: Dover, 1965.

[14] R. L. Rivest, A. Shamir, L. Adleman, "On digital signatures and public-key cryptosystems," Dep. Elec. Engr. and Comp Sci., M.I.T., Cambridge, MA, Tech. Rep. MIT/LCS/TM-82, Apr. 1977; also, to appear in *Comm of ACM* as "A method for obtaining digital signatures and public-key cryptosystems."

[15] N. J. A. Sloane, *A Handbook of Integer Sequences.* New York: Academic, 1973.

Optimal Source Coding for a Class of Integer Alphabets

PIERRE A. HUMBLET

***Abstract*—Let $p(i)$ be a probability measure on the set of nonnegative integers. The Huffman optimum encoding technique is extended to a class of $p(i)$ including those whose tail decreases**

Manuscript received August 5, 1976; revised May 2, 1977. This research was supported by the Advanced Research Projects Agency of the U.S. Department of Defense and was monitored by ONR, under Contract N00014-75-C-1183.

The author is with the Electronic Systems Laboratory, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139.

0018-9448/78/0100-0110$00.75 © 1978 IEEE