ACKNOWLEDGMENT

The authors thanks the anonymous referees and Associate Editor A. Barg who helped the authors improve their results and the exposition. In particular, one referee pointed out that the results are valid for all q; in the original manuscript, the authors only considered prime powers q and used $V = GF(q)$.

REFERENCES

[1] R. Ahlswede and G. Katona, "Contributions to the geometry of Hamming spaces," *Discrete Math.,* vol. 17, pp. 1–22, 1977.
[2] R. Ahlswede and I. Althöfer, "The asymptotic behaviour of diameters in the average," *J. Comb. Theory, Series B,* vol. 61, pp. 167–177, 1994.
[3] I. Althöfer and T. Sillke, "An 'average distance' inequality for large subsets of the cube," *J. Comb. Theory, Series B,* vol. 56, pp. 296–301, 1992.
[4] F.-W. Fu and S.-Y. Shen, "On the expectation and variance of Hamming distance between two binary i.i.d. random n-tuple," *Acta Mathematicae Applicatae Sinica,* vol. 13, no. 3, pp. 243–250, 1997.
[5] V. I. Levenshtein, "Bounds for packings of metric spaces and some of their applications," *Prob. Cyber.,* vol. 40, pp. 43–110, 1983 (in Russian).
[6] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.
[7] S. Roman, *Coding and Information Theory*. Berlin, Germany: Spring-Verlag, 1991.
[8] S.-T. Xia and F.-W. Fu, "On the average Hamming distance for binary codes," *Discrete Applied Mathematics,* to be published.

Note on Taking Square-Roots Modulo N

Eric Bach and Klaus Huber, *Member, IEEE*

***Abstract*— In this contribution it is shown how Gauss' famous cyclotomic sum formula can be used for extracting square-roots modulo N.**

***Index Terms*—Cryptography, factoring, Gauss sums, square-roots modulo N.**

I. INTRODUCTION

The task of taking square-roots modulo an integer N is a problem of considerable importance, in particular, as some well-known cryptographic schemes are based on this operation ([7], [25]). In this correspondence, we modify and analyze a famous formula due to Gauss (see [8], [12], and [15])

$$\sum_{s=0}^{n-1} e^{2\pi i s^2/n} = \begin{cases} (1+i)\sqrt{n}, & \text{for } n \equiv 0 \bmod 4 \\ \sqrt{n}, & \text{for } n \equiv 1 \bmod 4 \\ 0, & \text{for } n \equiv 2 \bmod 4 \\ i \cdot \sqrt{n}, & \text{for } n \equiv 3 \bmod 4 \end{cases} \tag{1}$$

to compute square-roots modulo N. We start with prime numbers and then comment on composite numbers. For simplicity without loss of generality the N considered are odd numbers. First, however, we give a short overview of other square-root methods.

Manuscript received December 21, 1997; revised July 6, 1998.

E. Bach is with the Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI 53706 USA.

K. Huber is with FE31a, Deutsche Telekom AG, Technologiezentrum, Am Kavalleriesand, Darmstadt, Germany.

Communicated by D. Stinson, Associate Editor on Complexity and Cryptography.

Publisher Item Identifier S 0018-9448(99)01389-9.

II. KNOWN METHODS FOR EXTRACTING SQUARE-ROOTS MODULO p

We consider the problem of finding the values of x in

$$x^2 \equiv n \bmod p \tag{2}$$

where n and p are given. Equation (2) has two solutions x_1 and $x_2 = -x_1$, which lie in $GF(p)$ if n is a quadratic residue, i.e., if the Legendre symbol $(n/p) = n^{(p-1)/2} \bmod p$ equals one.

Clearly, we have

$$n^{(p+1)/2} \equiv n \bmod p$$

and, thus, the determination of the roots $x_{1/2}$ is particularly easy if $(p+1)/2$ is even, i.e., for $p \equiv 3 \bmod 4$. In this case (see [3] and [11]), we immediately obtain the two solutions

$$x_{1/2} = \pm n^{(p+1)/4} \bmod p \quad \text{for } p \equiv 3 \bmod 4 \tag{3}$$

which can be determined efficiently using the square and multiply algorithm (see, e.g., [3], [10], and [20]).

For $(p+1)/2$ odd, i.e., $p \equiv 1 \bmod 4$, the situation is more difficult. In this case, all known efficient methods (see, e.g., [3], [6], [10], [16], [18], [20], [23], and [24]), are either probabilistic or make use of a quadratic nonresidue (QNR). Most of these are equivalent to or use the same basic ideas as either the Tonelli–Shanks or the Cipolla–Lehmer procedure.

The Tonelli–Shanks method [23], [24] relies on exponentiation in $GF(p)$. The method of Cipolla–Lehmer [6], [13], [20, pp. 287–288] uses exponentiation in $GF(p^2)$, which can be done efficiently using Lucas numbers. Compared to the Tonelli–Shanks method, the Cipolla–Lehmer method has the disadvantage that one has to determine a QNR which depends both on p and n, whereas the method of Tonelli–Shanks can reuse the same QNR for different n. From an aesthetic point of view this is an advantage if many square roots are computed with the same prime p, a case that often occurs in practice. From a computational point of view this advantage is not great as the Jacobi symbol algorithm is much faster than an exponentiation (see, e.g., [3, p. 113]). For further details on the computing time of Shanks' algorithms see [14].

In addition to the above two techniques, one can also use polynomial factoring methods such as the algorithms of Ben-Or/Rabin or Berlekamp (see [4], [5], and [19]). One probabilistic square-root method of Peralta [17, first algorithm] is a disguised version of Rabin/Ben-Or's algorithm (see [2, Remark 1, p. 1496]). It is interesting to note that a "shortcut" version of the Cipolla–Lehmer procedure does practically the same computation. In its polynomial version [3, p. 157] this procedure computes $\sqrt{n}$ as $X^{(p+1)/2}$ modulo the irreducible polynomial $X^2 - tX + n$. If we stop at the $(p-1)/2$ power, we obtain

$$X^{(p-1)/2} \equiv \sqrt{n} X^{-1} \equiv uX + v.$$

We have $X - t + nX^{-1} \equiv 0$, so $\sqrt{n} = -u^{-1}$.

Schoof's deterministic algorithm [22]—as the algorithm treated here—is efficient for small input values.

III. METHOD BASED ON GAUSS' FORMULA

Gauss' formula (1) can be modified in a rather straightforward way for use in finite fields $GF(p)$ by simply replacing the number

$e^{2\pi i/n}$ by an element γ of order n in the field $GF(p^m)$, where m is the smallest positive integer which fulfills $p^m \equiv 1 \bmod n$. Then m divides $\varphi(n)$, where $\varphi(n)$ is Euler's totient function (see, e.g., [9]). (More precisely, m divides $\lambda(n)$, the Carmichael function, see, e.g., [3] and [20].) Using the theory of cyclotomic fields we find that

$$\sum_{s=0}^{n-1} \gamma^{s^2} \equiv \begin{cases} (1+i)\sqrt{n} \bmod p, & \text{for } n \equiv 0 \bmod 4 \\ \sqrt{n} \bmod p, & \text{for } n \equiv 1 \bmod 4 \\ 0 \bmod p, & \text{for } n \equiv 2 \bmod 4 \\ i \cdot \sqrt{n} \bmod p, & \text{for } n \equiv 3 \bmod 4 \end{cases} \tag{4}$$

holds from which we immediately obtain the following explicit square-root formula for $n \not\equiv 2 \bmod 4$

$$\sqrt{n} \equiv \pm\kappa \cdot \sum_{s=0}^{n-1} \gamma^{s^2} \bmod p \tag{5}$$

where

$$\kappa = \begin{cases} \dfrac{1-i}{2}, & \text{for } n \equiv 0 \bmod 4 \\ 1, & \text{for } n \equiv 1 \bmod 4 \\ i, & \text{for } n \equiv 3 \bmod 4 \end{cases}$$

with $i^2 \equiv -1 \bmod p$. The case $n \equiv 2 \bmod 4$ can be reduced to the case $n \equiv 0 \bmod 4$ using $\sqrt{n} = \sqrt{4n}/2$.

Direct application of (5) is of course only practical for small values of n. In this case evaluation of (5) is particularly simple for $p \equiv 1 \bmod n$; as in this case, the element γ belongs to $GF(p)$. Also note, that for $p \equiv 1 \bmod 4$—for which (3) does not apply—the element $i = \sqrt{-1} \bmod p$ belongs to $GF(p)$ and can be expressed as a QNR of $GF(p)$ to the power of $(p-1)/4$.

Example 1: Consider the field $GF(p)$ with $p = 29$. We want to extract the square root of $n = 7$ modulo $p = 29$. As $29 \equiv 1 \bmod 7$ we can determine a suitable γ, e.g., by raising the primitive element 2 to the fourth power. Hence, we get $\gamma = 2^4 = 16$. Thus,

$$\sqrt{7} \equiv \pm i \sum_{j=0}^{6} \gamma^{j^2}.$$

The value $i = \sqrt{-1} \bmod p$ can easily be found as $i = 2^{(p-1)/4} = 12$, which leads to

$$\begin{aligned} \sqrt{7} &\equiv \pm 12 \cdot (1 + 16 + 16^4 + 16^9 + 16^{16} + 16^{25} + 16^{36}) \\ &\equiv \pm 6 \bmod 29. \end{aligned}$$

If carefully coded, the Gauss sum algorithm will be preferable to other known methods when $p \equiv 1 \bmod 8$ and n is a small odd prime dividing $p-1$. (By quadratic reciprocity, such n are always squares mod p.) We do the following:

1) choose a random QNR $\alpha \in GF(p)^*$;
2) compute $\beta = \alpha^{(p-1)/4n}$ and $i = \beta^n$ (we have $i = \sqrt{-1}$);
3) set $\gamma = \beta^4$; if this equals one go to 1);
4) compute

$$\sqrt{n} \equiv \pm\kappa \left(1 + 2\sum_t \gamma^t\right)$$

with the summation index t running over the nonzero squares mod n.

(Clearly, the formula in 4) is equivalent to (5) since there in the exponent each nonzero value occurs twice.) We now analyze the expected cost of this procedure. The first three steps use work equivalent to one exponentiation in $GF(p)$. At this point, i has been determined and $\gamma = 1$ with probability $1/n$. Therefore, γ and i can be determined at an average cost of

$$\frac{n}{n-1} \le \frac{3}{2}$$

exponentiations. We can compute the summation in the last step with about n multiplications in $GF(p)$, by listing $1, \gamma, \gamma^2, \cdots, \gamma^{n-1}$ and filtering out the powers coming from nonsquares.

All of the square-root algorithms we have described in Section II take time equivalent to at least two exponentiations in $GF(p)$, when $p \equiv 1 \bmod 8$. Writing E for an exponentiation time and M for a multiplication time, the Gauss sum method is preferable when $nM + (3/2)E < 2E$; that is, when $n < E/(2M)$. For all known exponentiation algorithms $E/M = O(\log p)$. We conclude that the Gauss sum algorithm should be used when $p \equiv 1 \bmod 8$ and $n = O(\log p)$. As numbers of size $O(\log p)$ can be factored with less time than E, we also see that the restriction of n being prime can be dropped, i.e., the method is efficient for $n = O(\log p)$ with n a divisor of $p-1$.

On the other hand, the method is only of theoretical interest when extensions to $GF(p)$ are required. In this case, it can be shown that our algorithm uses $O((n \log p)^4)$ bit operations, if ordinary arithmetic is employed. (The bottleneck is in constructing $GF(p^m)$—see [3, p. 172].) For fixed n this is comparable to Tonelli–Shanks, but worse than Cipolla–Lehmer.

Schönheim [21] used the Gauss sum to compute the square root of $\pm n \bmod p$, using a power of two as an nth root of unity mod p. He did not consider the running time nor the possibility of using other bases besides two. Further in Atkin [1] the method is essentially addressed briefly also without run-time analysis.

IV. Composite N

For composite N it is well-known that a square root of a number n modulo N can efficiently be extracted if the factorization of N is known. One simply computes the square roots of n modulo the prime-power factors of N and composes the result via the Chinese Remainder Theorem. If a prime p occurs with multiplicity e in N, then a formula given by Tonelli [24] applies. Namely, if y is a solution of $y^2 \equiv a \bmod p$, then a solution of $x^2 \equiv a \bmod p^e$ is given by

$$x \equiv y^{p^{e-1}} \cdot a^{(p^e - 2p^{e-1}+1)/2} \bmod p^e.$$

In practice, this formula can be replaced by the quadratically convergent form of Hensel's Lemma. Tonelli uses an exponentiation mod p^e, whereas Hensel's Lemma runs within a constant factor of a multiplication mod p^e (see [3, Theorem 7.7.1]). Alternatively, if n and N have no common divisor, (5) also holds for composite N

$$\sqrt{n} \equiv \pm\kappa \cdot \sum_{s=0}^{n-1} \gamma^{s^2} \bmod N \quad \text{for } \gcd(n, N) = 1. \tag{6}$$

Here γ is an element of multiplicative order exactly n in Z_N^* and modulo the prime factors of N.

V. Application to Factor Integers

Let $N = p_1 \cdots p_r$ be a product of odd primes, with n an odd prime divisor of $p_k - 1$ for all k. It is well known that if we are given γ, δ of order n in the multiplicative group mod N, neither a power of the other in that group, we can split N by computing the greatest common divisors (GCD)

$$\gcd(\gamma - \delta^k, N)$$

for $k = 0, \cdots, n-1$. On average this will use about $n/(r+1)$ GCD computations. We can do the job more quickly in many cases using the Gauss sum. Let γ and δ have order n modulo all p_k. Observe that if $(j/n) = -1$, we have

$$\sum_{s=0}^{n-1} \gamma^{s^2} + \sum_{s=0}^{n-1} (\gamma^j)^{s^2} = 2\left(\sum_{t=0}^{n-1} \gamma^t\right) \equiv 0 \pmod{p_k}$$

but the sums are equal when $(j/n) = +1$. [To see this, consider the possible values of (t/n).] If

$$\gamma = (\gamma_1, \cdots, \gamma_r)$$

and

$$\delta = (\gamma_1^{j_1}, \cdots, \gamma_r^{j_r})$$

—here we use "component" notation coming from the Chinese Remainder Theorem—then (abusing notation slightly)

$$S := \sum_{s=0}^{n-1} \gamma^{s^2} = (\sqrt{n}/\kappa_1, \cdots, \sqrt{n}/\kappa_r)$$

and

$$T := \sum_{s=0}^{n-1} \delta^{s^2} = (\pm\sqrt{n}/\kappa_1, \cdots, \pm\sqrt{n}/\kappa_r).$$

With probability $1/2^{r-1}$, the signs will not all agree, and $\gcd(S/T - 1, N)$ will split N. If the Gauss sums are computed as explained in Section III, then this uses about $2n$ multiplications and one gcd calculation.

Of course, for N consisting of large primes, the hard problem is to find suitable nth roots.

VI. Conclusion

An explicit formula for the extraction of square-roots modulo an integer N has been given using a famous sum formula of Gauss. The formula is interesting for theoretical purposes and for the extraction of square roots of small elements of Z_N. It is preferable to known methods for prime $N = p$ when $p \equiv 1 \bmod 8$ and n is a small odd prime dividing $p - 1$.

Acknowledgment

The authors would like to thank one referee for supplying reference [1].

References

[1] A. O. L. Atkin, "Probabilistic primality testing," summary François Morain, INRIA Res. Rep. 1779, pp. 159–163, Oct. 1992.

[2] E. Bach, "A note on square roots in finite fields," *IEEE Trans. Inform. Theory,* vol. 36, pp. 1494–1498, Nov. 1990.

[3] E. Bach and J. Shallit, "Algorithmic number theory," *Efficient Algorithms.* Cambridge, MA: MIT Press, 1996, vol. I.

[4] M. Ben-Or, "Probabilistic algorithms in finite fields," in *Proc. IEEE Symp. Found. Comp. Sci.,* 1981, pp. 394–398.

[5] E. R. Berlekamp, "Factoring polynomials over large finite fields," *Math. Comp.,* vol. 24, pp. 713–735, 1970.

[6] M. Cipolla, "Un metodo per la risolutione della congruenza di secondo grado," *Rendiconto dell'Accademia Scienze Fisiche e Matematiche,* Napoli, Ser. 3, vol IX, pp. 154–163, 1903.

[7] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Proc. Advances in Cryptology-Crypto.* New York: Springer-Verlag, 1987, pp. 186–194.

[8] C. F. Gauss, "Untersuchungen über Höhere Arithmetik," in *Disquisitiones Arithmeticae*, H. Maser, Ed. New York: Chelsea, 1889; 2nd reprint, 1981 (German transl.).

[9] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, 5th ed. Oxford, U.K.: Oxford, 1979.

[10] D. E. Knuth, "The Art of Computer Programming," *Seminumerical Algorithms*, 2nd ed. Reading, MA: Addison Wesley, 1981, vol. 2.

[11] J. L. Lagrange, "Sur la solution des problèmes indéterminés du second degré," *Histoire Acad. Roy. Sci. Belle-Lett.*, Berlin, Germany, 1769, pp. 165–310 (reprint Oeuvres, vol. 2, pp. 377–535).

[12] E. Landau, "Vorlesungen über Zahlentheorie, Aus der Elementaren Zahlentheorie," Hirzel 1927, Vierter Teil, Kapitel 6 Gaußsche Summen, reprint, Chelsea, New York, 1950.

[13] D. H. Lehmer, "Computer technology applied to the theory of numbers," in *Studies in Number Theory.* Englewood Cliffs, NJ: Prentice-Hall, 1969, pp. 117–151.

[14] S. Lindhurst, "An analysis of Shanks's algorithm for computing square roots in finite fields," in *Proc. 5th Conf. Canadian Number Theory Assoc.*, 1995.

[15] R. Lidl and H. Niederreiter, *Finite Fields, Encyclopedia of Mathematics and its Applications.* Cambridge, U.K.: Cambridge Univ. Press, 1984, vol. 20.

[16] J. L. Massey, "Cryptography: Fundamentals and applications," copies of transparencies, in *Adv. Technol. Seminars*, Zürich, Switzerland, pp. 6.42–6.44.

[17] R. Peralta, "A simple and fast probabilistic algorithm for computing square roots modulo a prime number," *IEEE Trans. Inform. Theory,* vol. IT-32, pp. 846–847, Nov. 1986.

[18] H. C. Pocklington, "The direct solution of the quadratic and cubic binomial congruences with prime moduli," in *Proc. Cambridge Phil. Soc.,* Cambridge, U.K., 1917, vol. 19, pp. 57–59.

[19] M. O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, pp. 273–280, 1980.

[20] H. Riesel, *Prime Numbers and Computer Methods for Factorization.* Boston, MA: Birkhäuser, 1985.

[21] I. Schönheim, "Formule pentru rezolvarea congruenţei $x^2 \equiv a \pmod P$ în cazuri pînă acum necunoscute si aplicarea lor pentru a determina direct rădăcinile primitive ale unor numere prime," *Acad. R. P. Rômine Fil. Cluj. Stud. Cerc. Mat. Fiz.*, vol. 7, 1956, pp. 51–58(, in Romainian, with Russian, French summaries).

[22] R. Schoof, "Elliptic curves over finite fields and the computation of square roots mod p," *Math. Computat.,* vol. 44, no. 170, pp. 483–494, Apr. 1985.

[23] D. Shanks, "Five number-theoretic algorithms," in *Proc. 2nd Manitoba Conf. Numer. Math.,* Manitoba, Canada, 1972, pp. 51–70.

[24] A.Tonelli, "Bemerkung über die Auflösung quadratischer Congruenzen," *Göttinger Nachrichten*, pp. 344–346, 1891.

[25] H. C. Williams, "A modification of the RSA public-key encryption procedure," *IEEE Trans. Inform. Theory,* vol. IT-26, pp. 726–729, 1980.