combined to form the sets of sequences. This proceeding reduces the numbers of generated sets of sequences extremely.

3) An efficient algorithm for checking sets of sequences to be a PGS builds the sum of all first PACF sidelobes and checks it to be zero. If this sum is zero the next PACF sidelobes are calculated and checked. Otherwise the set of sequences is not a PCS and the algorithm stops.

Example 6—Counterexample for Possible PCS: A counterexample is given for illustrating the reduction of the number of generated sets of $P=4$ sequences with $N=15$ elements. A simple binary counter would generate

$$2^{P\cdot N}=2^{60}\simeq 1.15\cdot 10^{18} \tag{63}$$

different sets of sequences that have to be checked.

However, with the equations (26), (38), and (40) only for sequences with $pl(a_0)=4$, $pl(a_1)=6$, $pl(a_2)=7$ and $pl(a_3)=7$, or with $pl(a_0)=5$, $pl(a_1)=5$, $pl(a_2)=6$, and $pl(a_3)=7$ PCS may exist. For $pl=4$ only 56 sequences, for $pl=5$ only 111 sequences, for $pl=6$ only 185 sequences and for $pl=7$ only 232 sequences with different PACF are obtained. So the number of all possible sets of sequence reduces to

$$56\cdot 185\cdot 232\cdot 232+111\cdot 111\cdot 185\cdot 232\simeq 83\cdot 1.09\cdot 10^{9}. \tag{64}$$

This number of possible sets can be checked for a PSC.

With these methods of computer search PCS with $P=3$ sequences could be found up to length $N=8$, with $P=4$ sequences up to length $N=19$ and with $P=5$ sequences up to length $N=12$. One example of these PCS are depicted in Fig. 2.

VII. Conclusion

In this correspondence sets of periodic complementary binary sequences (PCS) are examined. Properties and existence conditions are derived for PCS. The properties and synthesizing methods of the well-known sets of the aperiodic complementary binary sequences (ACS) could be generalized. Whereas ACS exist only for even number of sequences, PCS may exist for every number of sequences.

It is shown that PCS correspond to a subclass of difference families. Perfect binary arrays, whose two-dimensional periodic autocorrelation function is a delta function, yield a special subclass of PCS. A construction method based on perfect binary arrays is given.

Applying specified theorems recursively to known PCS produces PCS with increasing lengths and number of sequences. For PCS with length $N=0 \bmod 4$ the existence conditions do not restrict any number of sequences. With the described methods of computer search, PCS with four sequences for many lengths could be found. A construction method for PCS with four sequences and any number of elements is not known. A diagram provides a general view of all PCS up to length 50 and up to 12 sequences. Small PCS's obtained by methods of computer search are depicted in a table.

Acknowledgment

The authors acknowledge the referee, who pointed out the relation of PCS with difference families.

References

[1] M. Golay, "Complementary series," *IEEE Trans. Inform. Theory*, vol. IT-17, pp. 82–87, 1960.
[2] C. Tseng and C. Liu, "Complementary sets of sequences," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 644–665, 1972.
[3] E. Fenimore and T. Cannon, "Coded aperture imaging with uniformly redundant arrays," *Appl. Optics*, vol. 17, pp. 337–347, 1978.
[4] S. Golomb and H. Taylor, "Two-dimensional synchronization patterns for minimum ambiguity," *IEEE Trans. Inform. Theory*, vol. IT-28, pp. 600–604, 1982.
[5] J. Hersheya and R. Yarlagadda, "Two-dimensional synchronisation," *Electron. Lett.*, vol. 19, pp. 801–803, 1983.
[6] D. Calabro and J. Wolf, "On the synthesis of two-dimensional arrays with desirable correlation properties," *Inform. Contr.*, vol. 11, pp. 537–560, 1968.
[7] Y. Chan, M. Siu, and P. Tong, "Two-dimensional binary arrays with good autocorrelation," *Inform. Contr.*, vol. 42, pp. 125–130, 1979.
[8] L. Bömer and M. Antweiler, "Perfect binary arrays with 36 elements," *Electron. Lett.*, vol. 23, pp. 730–732, 1987.
[9] H. Lüke, "Sequences and arrays with perfect periodic correlation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-24, pp. 287–294, 1988.
[10] J. Jedwab and C. Mitchell, "Constructing new perfect binary arrays," *Electron. Lett.*, vol. 24, pp. 650–652, 1988.
[11] L. Bömer and M. Antweiler, "Two-dimensional perfect binary arrays with 64 elements," *IEEE Trans. Inform. Theory*, vol. 36, no. 2, pp. 411–414, Mar. 1990.
[12] P. Wild, "Infinite families of perfect binary arrays," *Electron. Lett.*, vol. 24, pp. 845–847, 1988.
[13] H. Lüke, L. Bömer, and M. Antweiler, "Perfect binary arrays," *Signal Processing*, vol. 17, pp. 69–80, 1989.
[14] J. Jedwab, C. Mitchell, F. Piper, and P. Wild, "Perfect binary arrays and difference sets," internal memo and private communication.
[15] G. Weathers and E. Holiday, "Group-complementary array coding for radar clutter rejection," *IEEE Trans. Aerosp. Electron. Syst.*, vol. AES-29, pp. 369–379, 1983.
[16] L. Baumert, *Cyclic Difference Sets*. Berlin: Springer Verlag, 1971.
[17] T. Beth, D. Jungnickel, and H. Lenz, *Design Theory*. Zürich: Bibliographisches Institut, 1985.

A Note on Square Roots in Finite Fields

ERIC BACH

***Abstract*—A simple method is presented showing the quadratic character in a finite field of odd order q can be computed in $O(\log^2 q)$ steps. It is also shown how sequences generated deterministically from a random seed can be used reliably in a recent randomized algorithm of Peralta for computing square roots in finite fields.**

I. Introduction

In [17], Peralta has published two interesting randomized algorithms for computing square roots modulo an odd prime p. In fact, his algorithms work in general finite fields, and in connection with this, two questions are of interest.

1) What is the best way to decide whether an element in a finite field is a square?
2) How should one choose successive pseudo-random numbers for use in these algorithms?

The first question arises because several randomized algorithms for computing square roots in a finite field of order q fail when a randomly constructed field element is a square. If failure can be predicted in less then $O(\log^3 q)$ steps—the time needed to run the algorithm—then another trial can be started with very little loss of time. One can test whether a nonzero element

Manuscript received October 18, 1988; revised March 27, 1990. This work was supported by the National Science Foundation, via grant CCR-8552596.
The author is with the Computer Sciences Department, University of Wisconsin, Madison, WI 53706.
IEEE Log Number 9037502.

t is a square by computing $t^{(q-1)/2}$, but this requires an $O(\log^3 q)$ computation. Section II gives another test that requires $O(\log^2 q)$ steps, which generalizes the Jacobi symbol algorithm to any finite field. The algorithm is implicit in the reciprocity law for polynomials over finite fields [12], but it does not seem to have been analyzed before.

The second question has already been studied in connection with some algorithms for computing square roots mod p. In [2] we showed that these algorithms are very reliable when run on sequences derived in a simple fashion from a randomly chosen seed. The results therein do not apply to Peralta's second algorithm, nor do they apply to non-prime finite fields. It therefore seemed interesting to ask whether these results could be extended to Peralta's algorithms. Section III reviews the algorithms, and Section IV answers this question in the affirmative.

II. A Quadratic Character Algorithm

In the sequel p denotes an odd prime, and $q = p^n$, a prime power. $\mathbb{F}_q$ or K denotes a finite field of q elements, K^* its group of nonzero elements, $\bar{K}$ its algebraic closure, and $K[X]$ the polynomial ring in one indeterminate. We assume that $\mathbb{F}_q$ is implemented as $\mathbb{F}_p(\xi)$, where ξ satisfies a monic polynomial equation of degree n over $\mathbb{F}_p$. Then in $\mathbb{F}_q$, addition and subtraction take $O(\log q)$ steps, and multiplication and division take $O(\log^2 q)$ steps (assuming that classical algorithms are used for integer and polynomial arithmetic). χ denotes the usual quadratic character on K; the following definition generalizes the Jacobi symbol to polynomials over finite fields.

Definition 2.1: Let K be a finite field of odd characteristic, and let f and g be polynomials in $K[X]$, with g monic and of positive degree. If g is irreducible, then

$$(f|g) = \begin{cases} 1, & \text{if } \gcd(f,g) = 1 \text{ and } f \text{ is a square mod } g; \\ -1, & \text{if } \gcd(f,g) = 1 \text{ and } f \text{ is not a square mod } g; \\ 0, & \text{otherwise.} \end{cases}$$

If $g = g_1 \cdots g_r$ with each g_i monic and irreducible, then $(f|g) = \prod_{i=1}^{r} (f|g_i)$.

This definition implies that if $\alpha \in K$, $(\alpha|g) = \chi(\alpha)^{\deg g}$, which can be proved as follows. It is enough to consider the case where g is irreducible, in which case $L = K[X]/(g)$ is an extension field of K, of degree $\deg g$. If α is not a square in K, then it becomes a square in L if and only if L contains a quadratic extension of K, that is, if and only if $\deg g$ is even.

The analog of quadratic reciprocity is the following result, for which an elegant proof is given by Ore [16, p. 272].

Theorem 2.2 [12]: Let $f, g \in K[X]$, with f, g monic and $\deg f, \deg g > 0$. Then

$$(f|g) = (-1)^{((|K|-1)/2)\deg f \deg g} (g|f).$$

Algorithm 2.3: Input—$f, g \in K[X]$, with g monic and $\deg g > 0$; output—$(f|g)$.

$f \leftarrow f \bmod g$.
$\alpha \leftarrow$ leading coefficient of f.
If $\deg f = 0$ then return $\chi(\alpha)^{\deg g}$.
$f \leftarrow f/\alpha$.
Return $\pm \chi(\alpha)^{\deg g}(g|f)$ (taking '$-$' iff $(|K|-1)/2$, $\deg f$, and $\deg g$ are all odd).

Theorem 2.4: If f and g in $K[X]$ have degree at most d, then Algorithm 2.3 computes $(f|g)$ using $O(d^2)$ field operations and $O(d)$ evaluations of the quadratic character in K. Consequently, the quadratic character in $\mathbb{F}_q$ can be evaluated with $O(\log^2 q)$ bit operations.

Proof: The correctness of the algorithm follows from the reciprocity law. To analyze its running time, note that it computes a polynomial remainder sequence

$$\begin{aligned} u_0 &= q_0 u_1 + \alpha_0 u_2 \\ u_1 &= q_1 u_2 + \alpha_1 u_3 \\ &\cdots \\ u_{k-1} &= q_{k-1} u_k + \alpha_{k-1} \end{aligned}$$

where $u_0 = f$, $u_1 = g$, u_i is monic for $i \geq 1$, and $\deg u_i > \deg u_{i+1}$ for $i = 1, \cdots, k-1$. Such remainder sequences have length $O(d)$ and can be computed with $O(d^2)$ field operations [15, p. 133]; this proves the first assertion.

To prove the second, note that if $\mathbb{F}_q = \mathbb{F}_p(\xi)$ where ξ has a monic defining polynomial g, then $\mathbb{F}_q \cong \mathbb{F}_p[X]/g(X)$, so $\chi(f(\xi)) = (f|g)$. In $\mathbb{F}_p$, χ can be evaluated in $O(\log^2 p)$ steps by the Jacobi symbol algorithm [6], and arithmetic operations take $O(\log^2 p)$ steps. So if $q = p^n$, the total number of bit operations used is at most a constant times

$$n^2(\log p)^2 + n(\log p)^2 = O\left(n^2 \log^2 p\right) = O\left(\log^2 q\right). \quad \square$$

Remarks:

1) The subresultant algorithm can be used to compute a remainder sequence that differs from $u_0, \cdots, u_k$ only by constant factors [15], from which it is easy to recover the u_i's and the α_i's. Then

$$(f|g) = \chi\left(\prod_{\substack{0 \leq i < k \\ \deg u_{i+1} \text{ odd}}} \alpha_i\right)(-1)^{((|K|-1)/2)\sum_{i=1}^{k-1} \deg u_i \deg u_{i+1}},$$

from which the quadratic character on $\mathbb{F}_q$ can be quickly reduced in parallel to arithmetic in $\mathbb{F}_p$ and one evaluation of χ. This gives another proof of Fich and Tompa's result [8] that for fields of small characteristic, the quadratic character is in the complexity class *NC*.

2) The algorithm given in this section can be improved asymptotically, to compute the quadratic character in $\mathbb{F}_q$ using $(\log q)^{1+o(1)}$ steps. We indicate briefly how this can be done.

Gauss [9, p. 509] showed how to compute the quadratic character in $\mathbb{F}_p$ using Euclid's algorithm; his procedure may be summarized as follows. Assume that a is positive and relatively prime to p. The Euclidean algorithm applied to a and p will compute quotients c_i and remainders v_i, where $v_0 = a$, $v_1 = p$, and $v_i = c_i v_{i+1} + v_{i+2}$ for $i = 0, \cdots, l-1$. (Then $v_l = 1$, $v_{l+1} = 0$, and $c_0, \cdots, c_{l-1}$ are quotients of the ordinary continued fraction of a/p.) For convenience, abbreviate $\lfloor v/2 \rfloor$ as v', and let

$$\psi = \sum_{i=0}^{l-1} \left\{ c_i \binom{v'_{i+1}+1}{2} + v'_{i+1} v'_{i+2} \right\}.$$

Then $(a|p) = \pm(-1)^{\psi}$, taking the plus sign if a is odd or $p \equiv \pm 1$ modulo 8, and the minus sign otherwise. (If p is

composite and a is relatively prime to p, this computes the Jacobi symbol [3, p. 290].)

Using this, $(a|p)$ can be computed in $(\log p)^{1+o(1)}$ steps by using Schönhage's algorithm [19] to find the continued fraction of a/p, then computing the remainders v_i modulo 8, and ψ modulo 2.

If q is not prime, then the formula of the previous remark may be used provided that the values of $\deg u_i$ and α_i can be found in $(\log q)^{1+o(1)}$ steps. This can be done via Moenck's algorithm [14], as follows. Moenck's algorithm finds the continued fraction quotients and greatest common divisor of the input polynomials, which determine the leading coefficient and degree of each polynomial in the ordinary remainder sequence. There is a relation expressing each such polynomial as a linear combination of the next two in the sequence; dividing this relation by the leading coefficient of that polynomial determines α_i. Since Moenck's algorithm uses $O(n \log^2 n)$ field operations on inputs of degree bounded by n, and each operation in $\mathbb{F}_p$ can be done in $(\log p)^{1+o(1)}$ steps [1, p. 277], the claimed bound follows.

3) Itoh and Tsujii [11] have given another algorithm for quadratic residuacity in $\mathbb{F}_q$. If $q = p^n$, their algorithm takes $O(n^3 \log^2 p)$ steps using classical methods, and by the previous results $O(n^\omega (\log p)^{1+o(1)})$ steps asymptotically, where ω is the exponent of matrix multiplication. (It is known that $2 \le \omega < 2.376$ [7].) In general, the algorithms of this section are faster, although the two results are comparable when n is fixed.
4) Even for prime fields, it seems to be unknown whether one can recognize dth powers in $\mathbb{F}_q$ in $O(\log^2 q)$ steps, when $d \neq 2$, and $d | q-1$.
5) The question of this section becomes moot if K has characteristic 2, for then every element is a square.

III. Two Square Root Algorithms

Below we review Peralta's results generalized to finite fields that have odd characteristic but are not necessarily prime. In the algorithm descriptions, T is an indeterminate, so that $K[T]$ is a polynomial ring.

Algorithm 3.1: Input—a, a nonzero square in K; output—a square root of a.

Choose $x \in K$ at random.
If $x^2 = a$, output x.
Otherwise, let $A = K[T]/(T^2 - a)$.
Find $u, v \in K$ such that in A, $(T + x)^{(q-1)/2} = uT + v$.
If $v = 0$, output u^{-1}.

Theorem 3.2 [17]: Algorithm 3.1 returns a square root of a unless $\chi(x^2 - a) = 1$. It takes $O(\log^3 q)$ steps, and fails with probability $1/2 - 3/(2q)$.

Algorithm 3.3: Input—a, a nonzero square in K; output—a square root of a.

Let $q - 1 = 2^e d$, where d is odd.
Choose $x \in K$ at random.
If $x = 0$ or $x^2 = -a$, fail.
Otherwise, let $A = K[T]/(T^2 + a)$.
Find $u, v \in K$ such that in A, $(T + x)^d = uT + v$.
If $uv = 0$, fail.
Otherwise, find the least i such that $(uT + v)^{2^i} = wT$ for some $w \in K$.
Define $r, s \in K$ by $(uT + v)^{2^{i-1}} = rT + s$.
Return s/r.

Theorem 3.4 [17]: Let b denote a square root of $-a$, and let $m = 2^{e-1}$; assume $q \equiv 1 \pmod 4$. Algorithm 3.3 returns a square root of a unless $x = \pm b$ or for some nonzero y, $(x + b)/(x - b) = y^m$. It takes $O(\log^3 q)$ steps, and fails with probability at most $1/m + 1/q$.

Ordinarily one simply repeatedly tries a randomized algorithm until it works. However, the results of Section II suggest the following improvement to Algorithm 3.1: Test random values of x until one is found with $\chi(x^2 - a) \neq 1$, rather than repeat the whole algorithm. (This strategy was suggested by Berlekamp [4].)

If m is large, Algorithm 3.3 is much more reliable than Algorithm 3.1. Unfortunately, there seems to be no better way to tell whether a choice of x will work than to try it. However, any x for which $\chi(x^2 + a) = -1$ will succeed in Algorithm 3.3, for then $(x + b)/(x - b)$ is not a square, and surely not an mth power. One might also use Algorithm 2.3 here to quickly find such an x.

Remarks:

1) Peralta's first method is a simplified version of the Berlekamp–Rabin algorithm for factoring $T^2 - a$. For, if $(T + x)^{(q-1)/2} \equiv uT \bmod (T^2 - a)$, then Rabin's algorithm [18] would compute $\gcd((T + x)^{(q-1)/2} - 1, T^2 - a) = T - u^{-1}$. Since $K[T] \cong K[T + x]$, replacing $T + x$ by T shows that $\gcd(T^{(q-1)/2} - 1, (T - x)^2 - a) = T - (x + u^{-1})$, from which Berlekamp's algorithm [4] would return u^{-1} as a square root of a.
2) The failure criterion for Algorithm 3.1 is nearly identical to that of the Cipolla–Lehmer algorithm [13], which computes $\sqrt{a}$ in $O(\log^3 q)$ steps when a random $x \in K$ satisfies $\chi(x^2 - 4a) \neq 1$. Also, given $x \in K$ for which $\chi(x) = -1$, the Tonelli–Shanks algorithm [21] will compute a square root in K in $O(\log^4 q)$ steps. Both of these methods could profit from Algorithm 2.3.
3) If $q \equiv 3 \pmod 4$, Algorithm 3.3 can be used, provided that $q - 1$ is replaced by $q + 1$ in the first line. If $m = 2^{e-1}$, the computation fails iff $T + x$ is an mth power in A (a finite field of order q^2), which occurs with probability at most $1/m$.
4) All algorithms discussed in this section for computing square roots in K take at least $O(\log^3 q)$ steps. It is unknown whether this can be reduced to $O(\log^2 q)$, although when K has characteristic 2, one can compute square roots by inverting the matrix for the Frobenius map $x \to x^2$ [5]. This method takes $O(\log^2 q)$ steps once the inverse matrix is computed.

IV. Deterministic Sequences for Square Root Computation

The algorithms of the last section will in general require a sequence of random inputs from K; this section discusses simple methods to generate them from a random seed $x \in K$. The results show that if fixed constants $c_1, \cdots, c_k$ are properly chosen, then trials using $x + c_1, \cdots, x + c_k$ will simulate independence.

Definition 4.1: A *line* in $\mathbb{F}_q$ is a set of the form $\{\gamma + \delta t : t \in \mathbb{F}_p\}$, where $\gamma, \delta \in \mathbb{F}_q$ and $\delta \neq 0$.

In the results next, we assume that $c_1, \cdots, c_k$ are distinct elements of K, chosen from a set containing no lines (this is the

interpretation of "properly chosen"). Such sets are easy to find. For example (viewing $\mathbb{F}_q$ as a vector space over $\mathbb{F}_p$), $\{(x_1,\cdots,x_n)$: all $x_i \neq 0\}$ contains no lines. Also, any set of size less than p cannot contain a line.

Lemma 4.2: Let $m|q-1$, with $m>1$, and let $0<e_i<m$ for $i=1,\cdots,l$. Then

$$f(X)=\prod_{i=1}^{l}\left((X+c_i+b)(X+c_i-b)^{m-1}\right)^{e_i}$$

is not an mth power in $\bar{K}[X]$.

Proof: The zeros of f are $\alpha_i=b-c_i$ and $\beta_i=-b-c_i$. Certainly the α_i's are distinct, as are the β_i's. Hence no three of them can be equal. By unique factorization, if f is an mth power, then $\{\alpha_1,\cdots,\alpha_l\}=\{\beta_1,\cdots,\beta_l\}$, that is,

$$\{c_1-b,\cdots,c_l-b\}=\{c_1+b,\cdots,c_l+b\}.$$

Thus $C=\{c_1,\cdots,c_l\}$ is invariant under translation by $2b$, so C must be a disjoint union of sets of the form $\{\gamma+2bt\colon t\in\mathbb{F}_p\}$. This contradicts the hypothesis that C contains no lines. □

Lemma 4.3: Let $m|q-1$, with $m>1$, and let $b\neq 0$. If N denotes the number of $(x,y_1,\cdots,y_k)$ in K^{k+1} satisfying

$$(x+c_i+b)(x+c_i-b)^{m-1}=y_i^m,\qquad i=1,\cdots,k,$$

then $N\le q+2km^k\sqrt{q}$.

Proof: Let χ denote a character of order m on K^*. Weil showed that if $f\in K[X]$, with d distinct roots in $\bar{K}$, but not an mth power, then

$$\left|\sum_{x\in K}\chi(f(x))\right|\le(d-1)\sqrt{q}$$

[19, p. 43]. The number of solutions $(x,y)\in K^2$ to $y^m=f(x)$ is $1+\chi(f(x))+\cdots+\chi^{m-1}(f(x))$, so each $x\in K$ corresponds to

$$\prod_{i=1}^{k}\sum_{0\le e_i<m}\chi\left((x+c_i+b)(x+c_i-b)^{m-1}\right)^{e_i}$$

solutions $(x,y_1,\cdots,y_k)$ of the equations in the lemma. Therefore

$$N=\sum_{x\in K}\prod_{i=1}^{k}\sum_{0\le e_i<m}\chi\left((x+c_i+b)(x+c_i-b)^{m-1}\right)^{e_i}$$

$$=\sum_{0\le e_1,\cdots,e_k<m}\ \sum_{x\in K}\chi\left((x+c_1+b)(x+c_1-b)^{m-1}\right)^{e_1}$$

$$\cdots\chi\left((x+c_k+b)(x+c_k-b)^{m-1}\right)^{e_k}.$$

Group the terms in the first sum according to how many e_i's are nonzero and use Weil's estimate and Lemma 4.2 to show that N is at most

$$q+\sum_{i=1}^{k}(2i-1)(m-1)^i\binom{k}{i}\sqrt{q}$$

$$=q+\left(m^{k-1}[2k(m-1)-m]+1\right)\sqrt{q}\le q+2km^k\sqrt{q}.\quad\square$$

Theorem 4.4: Choose $x\in K$ at random. The probability that Algorithm 3.1 fails on $x+c_i$, $i=1,\cdots,k$, is at most $1/2^k+2k/\sqrt{q}$. If $k=\lfloor(\log_2 q)/2\rfloor$, this is $O(\log q/\sqrt{q})$.

Proof: If all trials fail, there are nonzero $y_1,\cdots,y_k\in K$ for which $(x+c_i)^2-a=y_i^2$, $i=1,\cdots,k$. That is, $(x+c_i+b)(x+c_i-b)=y_i^2$, $i=1,\cdots,k$, where $b^2=a$. These k equations have at most $q+k2^{k+1}\sqrt{q}$ solutions by Lemma 4.3, and each x for which all trials fail is associated with exactly 2^k such solutions because the signs of the y_i's may be chosen arbitrarily. Therefore the number of unsuccessful values of x is at most $q/2^k+2k\sqrt{q}$. Dividing by q gives the first result; the second follows by substitution. □

Theorem 4.5: Choose $x\in K$ at random. The probability that Algorithm 3.3 fails on $x+c_i$, $i=1,\cdots,k$, is at most $1/m^k+2k/\sqrt{q}+2k/q$. If $k=\lfloor(\log_m q)/2\rfloor$, this is $O(\log q/(\sqrt{q}\log m))$.

Proof: Choose b with $b^2=-a$. If all trials fail either $x=-c_i\pm b$ for some i or there are nonzero $y_1,\cdots,y_k\in K$ such that $(x+c_i+b)(x+c_i-b)^{m-1}=y_i^m$, $i=1,\cdots,k$. The rest of the proof is similar to that of Theorem 4.4. □

Remarks:

1) Theorems 4.4 and 4.5 can be sharpened, with a more technical proof. However, the improvement is slight—a factor of two at best—so the argument is only sketched. We assume that the reader is familiar with algebraic curves; the terminology follows Hartshorne [10, Chapter IV].

   First, let N' denote the number of solutions in K to the equations $(x+c_i+b)/(x+c_i-b)=y_i^m$, $i=1,\cdots,k$. These equations define an algebraic curve C. Any solution to these equations is nonsingular; C has other points (also nonsingular) that can be found by letting $x=1/t$, clearing fractions, and setting t to zero. By Weil's theorem [10, p. 368], $N'\le q+2g\sqrt{q}$, where g is the genus of the nonsingular curve C' associated with C.

   To estimate g, consider C' as a covering of the projective plane. This has degree m^k and is ramified only when some $x+c_i\pm b$ is zero. The ramification index can be computed by considering the case where $i=1$ and $x+c_1+b$ occurs only in the numerator of the first equation, as the other cases can be reduced to this one via birational transformations. Then $y_1^m\sim(x+c_1+b)$, and there are m^{k-1} distinct values for the y_i's when $x=-c_1-b$. Taking all possible x into account, there are at most $2km^{k-1}$ ramification points, each of index m. By Hurwitz's formula [10, p. 301], $2g-2\le m^k(-2)+2km^{k-1}(m-1)$, so $g\le km^k(m-1)$.

   Using this, one gets an estimate for N' that can be used in place of Lemma 4.3. If this is done, then the bound of Theorem 4.4 becomes $1/2^k+k/\sqrt{q}$, and that of Theorem 4.5 becomes $1/m^k+2k(m-1)/(m\sqrt{q})+2k/q$.
2) If $m\ge\sqrt{q}$, then Theorem 3.4 is sharper than Theorem 4.5.
3) Theorem 4.4 evidently applies to the Cipolla–Lehmer algorithm, and to the modification of Algorithm 3.3 suggested in Section IV. If one merely wants to find an element in K that is not an mth power (i.e., for use in the Tonelli–Shanks algorithm), then a bound similar to Theorem 4.5 holds, under the weaker assumption that $c_1,\cdots,c_k$ are distinct [22].

Acknowledgment

Thanks are due to Gilles Brassard for posing the question of Section IV, and to Victor Shoup for suggesting the use of character sums. Remark 2 of Section II follows suggestions by Jeffrey Shallit and Victor Shoup. Remark 3 of Section III was suggested by Don Coppersmith. Thanks also to the referees and other readers for many helpful comments.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Reading, MA: Addison-Wesley, 1974.
[2] E. Bach, "Realistic analysis of some randomized algorithms," in *Proc. Nineteenth Ann. ACM Symp. Theory of Comput.*, 1987, pp. 453–461. [Final version to appear, *J. Comput. Syst. Sci.*]
[3] P. Bachmann, *Niedere Zahlentheorie*, vol. 1. New York: Chelsea, 1968.
[4] E. R. Berlekamp, "Factoring polynomials over large finite fields," *Math. Comp.*, vol. 24, pp. 713–735, 1970.
[5] E. R. Berlekamp, H. Rumsey, and G. Solomon, "On the solution of algebraic equations over finite fields," *Inform. Contr.*, vol. 10, pp. 553–564, 1967.
[6] G. Collins and R. Loos, "The Jacobi symbol algorithm," *SIGSAM Bull.*, vol. 16, pp. 12–16, 1982.
[7] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," in *Proc. Nineteenth Ann. ACM Symp. Theory of Computing*, 1987, pp. 1–6. [Final version to appear, *J. Symbol. Comp.*]
[8] F. E. Fich and M. Tompa, "The parallel complexity of exponentiating polynomials over finite fields," *J. ACM*, vol. 35, pp. 651–667, 1988.
[9] C. F. Gauss, "Neue Beweise und Erweiterungen des Fundamentalsatzes in der Lehre von den quadratischen Resten," in C. F. Gauss, *Untersuchen über Höhere Arithmetik*. New York: Chelsea, 1965, pp. 496–510.
[10] R. Hartshorne, *Algebraic Geometry*. Berlin: Springer-Verlag, 1977.
[11] T. Itoh and S. Tsujii, "An efficient algorithm for deciding quadratic residuosity in finite fields GF(p^m)," *Inform. Processing Lett.*, vol. 30, pp. 111–114, 1989.
[12] H. Kühne, "Eine Wechselbeziehung zwischen Functionen mehrer Unbestimmten, die zu Reciprocitätsgesetzen führt," *J. Reine Angew. Math.*, vol. 124, pp. 121–131, 1902.
[13] D. H. Lehmer, "Computer technology applied to the theory of numbers," in *Studies in Number Theory*, W. J. LeVeque, Ed. Washington, D.C.: MAA, 1969.
[14] R. Moenck, "Fast computation of GCD's," *Proc. Fifth Ann. ACM Symp. Theory of Computing*, 1973, pp. 142–151.
[15] R. Loos, "Generalized polynomial remainder sequences," in *Computer Algebra: Symbolic and Algebraic Computation*, 2nd ed., B. Buchberger, G. E. Collins, and R. Loos, Eds. Wien: Springer-Verlag, 1983, pp. 115–137.
[16] O. Ore, "Contributions to the theory of finite fields," *Trans. Amer. Math. Soc.*, vol. 36, pp. 243–274, 1934.
[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, 1986.
[18] M. O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, pp. 273–280, 1980.
[19] W. M. Schmidt, *Equations over Finite Fields: An Elementary Approach*. Berlin: Springer-Verlag, 1976.
[20] A. Schönhage, "Schnelle Berechnungen von Kettenbruchentwicklungen," *Acta Informatica*, vol. 1, pp. 139–144, 1971.
[21] D. Shanks, "Five number-theoretic algorithms," in *Proc. Second Manitoba Conf. Numerical Math.*, 1972, pp. 51–70.
[22] V. Shoup, "New algorithms for finding irreducible polynomials over finite fields," *Math. Comp.*, vol. 54, pp. 435–447, 1990.

Correction to "On Normal and Subnormal q-ary Codes"

ANTOINE C. LOBSTEIN AND GERHARD J. M. VAN WEE

In the above correspondence,[1] the following corrections are necessary.

When sets are defined, a vertical bar | is intended where a division bar / is used. The most important place where this might cause confusion is in the proof of Lemma 1. A rewritten version of part of that proof will follow.

The last two sentences of the Introduction should read: We include a table of lower and upper bounds on $K_3(n, R)$, the minimal number of codewords in any ternary code of length n and covering radius R, for $n \le 13$, $R \le 3$, known to us. We improved some of the known lower bounds by linear programming.

Section II, line 13: ..., and such a coordinate i is called *acceptable*.

Proof of Theorem 1, line 5: $\cdots + d((u,v), B_a^{(1)}) - \Delta_{a,u}\}$.

Theorem 2 should read: If C is a $(q,n,M)R$ subnormal code with an acceptable partition without the empty set, then for every natural number p there is a $(q, n+pq, M)R + (q-1)p$ code.

In the proof of Lemma 1, the first few lines should read:

Proof: The repetition code is $C_{\text{rep}} = \cup_{a \in F_q}\{J_a^n\}$, with J_a^n the all-a vector of length n. Let w be any vector in F_q^n, containing p_a times the symbol a. Let $p = \max\{p_a | a \in F_q\}$. Then $p \ge \lfloor n/q \rfloor$ and $d(w, C_{\text{rep}}) = n - p \le n - \lfloor n/q \rfloor$ and so C_{rep} has covering radius $R \le n - \lfloor n/q \rfloor$. Taking w with $p = \lfloor n/q \rfloor$ shows that $R = n - \lfloor n/q \rfloor$. Now, ...

Three lines before Theorem 3 should read: ... are nonempty for all $a \in F_q$.

The second sentence of the proof of Theorem 3 should read: For $t \in F_q$ let $\Delta_t = 0$ if $t = 0$, and $\Delta_t = 1$ otherwise.

Two lines before Lemma 3, the name should read: J. H. van Lint, Jr.

On page 1293, first column, line 4: $\cdots + \Sigma_{a \in F_q \setminus \{c_1\}} d(x, b^a)$.

The middle of line 2 of Theorem 5 should read: then $d \le (q/(q-1)) \cdot R + 1$.

The first sentence in the proof of Theorem 5 should end with: $d(c, \varnothing) = n \ge d$.

The second to last sentence of Section III should read: Theorem 5 and any choice of the parameters of the Hamming codes just mentioned can be used to disprove the q-ary generalization of this conjecture, even when we replace "normal" by "subnormal."

On page 1293, second column, line 2 should read: $|C| \ge 3^n/(1+2n)$.

Proof of Theorem 6, line 3: ...such that $d(c, c') \le 2$.

Page 1294, second column, line 8 the C should be uppercase.

In Section V, Open Problem 1) should read:

1) Find ternary, optimal or nonoptimal, normal or subnormal codes improving, by the amalgamated direct sum construction, on the upper bounds on $K_3(n, R)$ (cf. Section IV-A).

The following piece of text is missing at the end of the paper.

Notes Added in Proof

1) The result, mentioned in the Introduction, that binary linear codes with minimum distance $d_{\min} \le 5$ are normal, has not (yet) been established. X. Hou (Univ. of Chicago) has shown that the proof in [12] is incorrect.
2) For open problem No. 2, see: G. J. M. van Wee, "Bounds on packings and coverings by spheres in q-ary and mixed Hamming spaces," *J. Combin. Theory* (*A*), to appear.

In [5], there are two authors, H. O. Hämäläinen and S. Rankinen. Reference [8] appeared in *IEEE Trans. Inform. Theory*, vol. IT-34, pp. 1343–1344, Sept. 1988.

Manuscript received May 1990.

A. C. Lobstein is with the Centre National de la Recherche Scientifique, URA 251, Télécom Paris, Département Informatique, 46 rue Barrault, 75634 Paris Cedex 13, France.

G. J. M. van Wee is with the Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands.

IEEE Log Number 9037999.

[1] A. C. Lobstein and G. J. M. van Wee, *IEEE Trans. Inform. Theory*, vol. 35, no. 6, pp. 1291–1295, Nov. 1989.