J. Cryptology (2005) 18: 337–355
DOI: 10.1007/s00145-004-0221-0

Journal of
CRYPTOLOGY

Polynomial and Normal Bases for Finite Fields

Joachim von zur Gathen
B-IT Computer Security,
Universität Bonn,
Görresstraße 13,
D-53113 Bonn, Germany
gathen@bit.uni-bonn.de

Michael Nöcker
Fakultät für Elektrotechnik,
Informatik, Mathematik,
Universität Paderborn,
D-33095 Paderborn, Germany
noecker@uni-paderborn.de

Communicated by Arjen K. Lenstra

Received 8 March 2002 and revised 7 July 2003
Online publication 21 April 2005

Abstract. We discuss two different ways to speed up exponentiation in nonprime finite fields: on the one hand, reduction of the total number of operations, and on the other hand, fast computation of a single operation. Two data structures are particularly useful: sparse irreducible polynomials and normal bases. We report on implementation results for our methods.

Key words. Exponentiation, Finite fields, Normal basis, Polynomial basis, Gauss period.

1. Introduction

This paper deals with fast exponentiation in finite fields $\mathbb{F}_{q^n}$, which is a fundamental operation in several cryptosystems (e.g., Diffie and Hellman, 1976; ElGamal, 1985). There are two different ways to speed up exponentiation: reducing the number of operations in $\mathbb{F}_{q^n}$, or improving each single operation. There is a particularly attractive data structure for finite fields, namely normal bases, which gives us qth powers essentially for free. The task then is to reduce the number and cost of (other) multiplications.

Our goal is to compare two well-known representations of $\mathbb{F}_{q^n}$: polynomial and normal bases. In Section 2 we study three approaches using a polynomial basis. Namely we discuss two types of sparse polynomials: *sedimentary polynomials*, which have all nonzero terms at low degrees, except the leading one, and the usual *sparse polynomials* with as few nonzero terms as possible, mainly *trinomials*. A third method uses the polynomial representation of the Frobenius automorphism introduced by von zur Gathen and Shoup (1992) and modular composition. In Section 3 we compare classical arithmetic for normal bases with the work of Gao et al. (2000) which connects normal bases and fast polynomial arithmetic using Gauss periods. The theoretical estimates for the better ones of these methods are too close to each other to distinguish between them. Therefore we ran a substantial series of experiments, reported in Section 4. Two champions emerge: sparse irreducible polynomials, in particular trinomials, for the polynomial representation and, when available, normal bases generated by Gauss periods of type $(n, 1)$.

2. Polynomial Basis

Let $q, n \in \mathbb{N}_{\geq 2}$ with q a prime power, and let $\mathbb{F}_{q^n}$ be the finite field with q^n elements. Regarding $\mathbb{F}_{q^n}$ as a vector space of dimension n over $\mathbb{F}_q$, we consider two different types of bases in this and the next section: polynomial and normal bases.

Let $f \in \mathbb{F}_q[x]$ be an irreducible polynomial of degree n. Then we have $\mathbb{F}_{q^n} \cong \mathbb{F}_q[x]/(f)$, and $((1 \bmod f), (x \bmod f), \ldots, (x^{n-1} \bmod f))$ is the canonical *polynomial basis*. The *canonical representative* of $\beta \in \mathbb{F}_{q^n}$ is the unique polynomial $g \in \mathbb{F}_q[x]$ of degree less than n such that $(g \bmod f) = \beta$. We call a function $\mathsf{M}\colon \mathbb{N}_{>0} \to \mathbb{R}_{>0}$ a *multiplication time* for $\mathbb{F}_q[x]$ if polynomials in $\mathbb{F}_q[x]$ of degree less than n can be multiplied using at most $\mathsf{M}(n)$ operations in $\mathbb{F}_q$. Classical polynomial multiplication yields $\mathsf{M}(n) \leq 2n^2$. We can choose $\mathsf{M}(n) \in O(n \log n \, \log\log n)$ according to Schönhage and Strassen (1971) and Schönhage (1977). Detailed presentations can be found in Section 8.3 of Aho et al. (1974), and in Chapter 9 of von zur Gathen and Gerhard (2003). Implementations are discussed in von zur Gathen and Gerhard (2002); the crossover between classical and Karatsuba multiplication is at degree 576 for that implementation. Allowing $O(\mathsf{M}(n))$ precomputation depending only on f, two elements of $\mathbb{F}_{q^n}$ can be multiplied with at most $3\mathsf{M}(n) + O(n)$ operations in $\mathbb{F}_q$. Using an appropriate addition chain for exponentiation (Brauer, 1939; see Section 4.6.3 of Knuth 1998) we get the following result.

Fact 2.1. *Let $e \in \mathbb{N}_{>0}$ with $2 \leq e < q^n$ and let $\mathbb{F}_{q^n}$ be represented by a polynomial basis. An element of $\mathbb{F}_{q^n}$ can be raised to the* eth *power with $3n\mathsf{M}(n) \log q + O(n^2 \log q) \subseteq O(n^2 \log n \log\log n \, \log q)$ operations in $\mathbb{F}_q$.*

Using *modular composition* à la Brent and Kung (1978) and the polynomial representation of the Frobenius automorphism from von zur Gathen and Shoup (1992), Gao et al. (2000) present an algorithm which uses $O(n^2 \log\log n)$ operations in $\mathbb{F}_q$.

Sparse modulus. We consider two kinds of *sparse* polynomials. An *s-sparse polynomial* f in the usual sense is of the form $f = \sum_{1 \leq i \leq s} f_i x^{e_i}$ with all $f_i \in \mathbb{F}_q \setminus \{0\}$ and $e_i \in \mathbb{N}_{\geq 0}$, and we want the number s of nonzero terms to be small. The minimal sparseness of irreducible polynomials of this kind is

$$\sigma_q(n) = \min \left\{ s \in \mathbb{N}_{>0} \colon \begin{array}{l} \text{there exists an } s\text{-sparse irreducible polynomial} \\ \text{in } \mathbb{F}_q[x] \text{ of degree } n \end{array} \right\}.$$

A special type of sparse polynomials is of the form $f = x^n + h \in \mathbb{F}_q[x]$ with $t = \deg h \ll n$ small. All the "relevant material" of f sits at the bottom, and we call f a *t-sedimentary polynomial*. For $n \in \mathbb{N}_{>0}$ and a prime power $q \geq 2$, we define

$$\tau_q(n) = \min\{\deg h \colon x^n + h \in \mathbb{F}_q[x] \text{ is irreducible}\}.$$

Obviously $\sigma_q(n) - 2 \leq \tau_q(n) \leq n - 1$, since sedimentarity is a special case of sparseness with $s \leq t + 2$.

Sparse polynomials. We first discuss arithmetic in $\mathbb{F}_q[x]/(f)$ for a sparse polynomial f. The following algorithm (which actually works over any commutative ring R) computes the quotient u and remainder v of a polynomial g on division by a monic sparse polynomial f. The idea is based on the observation that the top part of u, called u_1, equals the top part of g. Furthermore, the bottom part u_0 of u equals the top part of $g - u_1 f x^{k-n}$ for a suitable $k \geq n$; this yields a recursive approach.

Algorithm 2.2. Sparse division.

Input: An integer $n \in \mathbb{N}_{>0}$, a polynomial $g \in R[x]$ of degree m, and $f = \sum_{1 \leq i \leq s} f_i x^{e_i}$, where $0 = e_1 < \cdots < e_s = n$ and $f_1, \ldots, f_s \in R$ with $s \geq 2$ and $f_s = 1$, and R is a commutative ring.

Output: Uniquely determined polynomials $u, v \in R[x]$ such that $\deg v < n$ and $g = u \cdot f + v$.

1. If $m < n$ then
2. set $(u, v) \leftarrow (0, g)$. Return (u, v).
3. Set $k = \max\{n, m - (n - e_{s-1}) + 1\} \geq n$. Write $g = u_1 x^k + w$ with $u_1, w \in \mathbb{F}_q[x]$ and $\deg w < k$.
4. Compute $g_1 \leftarrow w - u_1 \cdot (f - x^n) x^{k-n}$.
5. Call the algorithm recursively with input n, g_1 and f to receive (u_0, v).
6. Set $u \leftarrow u_1 x^{k-n} + u_0$.
7. Return (u, v).

Theorem 2.3. *Algorithm* 2.2 *works correctly. Division with remainder of a polynomial of degree m by an s-sparse monic polynomial of degree n can be executed using at most* $2(s-1)(m-n+1)$ *operations in R (if $m + 1 \geq n$).*

Proof. We prove correctness by induction on m, and thus assume that the algorithm works correctly if the dividend has degree less than m. We always have $0 = e_1 \leq e_{s-1} < e_s = n$. Furthermore, in Step 3 we are working on the case that $n \leq m$. We find that $k = \max\{n, m - (n - e_{s-1}) + 1\} \leq \max\{n, m\} \leq m$ and thus $n \leq k \leq m$. By the induction hypothesis we get

$$\begin{aligned} uf + v &\overset{6.}{=} (u_1 x^{k-n} + u_0) f + v = u_1 f x^{k-n} + (u_0 f + v) \\ &\overset{5.}{=} u_1 f x^{k-n} + g_1 \overset{4.}{=} u_1 f x^{k-n} + w - u_1 (f - x^n) x^{k-n} \\ &= w + u_1 x^k + (u_1 f x^{k-n} - u_1 f x^{k-n}) \overset{3.}{=} g. \end{aligned}$$

Since $\deg v < n$, this shows partial correctness. In order to prove termination, we show that the degree of g_1 is less than $\deg g = m$. Since $\deg w < k$ and

$$\deg u_1 \ = \ \deg g - \deg(x^k) = m - k \geq 0,$$

we have

$$\begin{aligned}
\deg g_1 \ &\leq \ \max\{\deg w, \deg(u_1 \cdot (f - x^n)x^{k-n})\} \\
&\leq \ \max\{k - 1, m - k + e_{s-1} + k - n\} \\
&= \ \max\{\max\{n, m - (n - e_{s-1}) + 1\} - 1, m - (n - e_{s-1})\} \\
&= \ \max\{n, m - (n - e_{s-1}) + 1\} - 1 = k - 1 < m,
\end{aligned}$$

and the algorithm terminates.

Now we turn to the cost estimate. The polynomials u_1 and w can be generated in Step 3 from g without operations in $\mathbb{F}_q$. The cost to compute g_1 is given by subtracting the polynomial $f - x^n = \sum_{1 \leq i \leq s} f_i x^{e_i} - f_s x^{e_s} = \sum_{1 \leq i < s} u_1 f_i x^{e_i + (k-n)}$ from w in Step 4. For each $1 \leq i < s$ we can compute $u_1 f_i x^{e_i + (k-n)}$ with at most $1 + \deg u_1 = 1 + m - k$ scalar multiplications, and subtract it as required with $m - k + 1$ subtractions, so that we have a total of $2(s-1)(m-k+1)$ operations in $\mathbb{F}_q$. Since $g_1 = u_0 \cdot f + v$ and hence $\deg u_0 = \deg g_1 - \deg f \leq k - 1 - n < k - n$, no more operations in $\mathbb{F}_q$ are needed to compute u.

If $T(m)$ denotes the number of operations in $\mathbb{F}_q$ to compute (u, v) for g of degree m, then we have $T(m) = 0$ if $m < n$ and $T(m) \leq 2(s-1)(m-k+1) + T(k-1)$ since $\deg g_1 \leq k - 1$. A recursive call of Algorithm 2.2 in Step 5 reduces the remaining problem size by $m - k + 1$, namely from m to $k - 1$. For the last call we have $k = n$. We have a total of $S = \lceil (m - n + 1)/(n - e_{s-1}) \rceil$ calls. All but the last call are performed with $2(s-1)(n - e_{s-1})$ operations. The last call causes $2(s-1)(m - n + 1 - (S-1) \cdot (n - e_{s-1} + 1))$ operations. This yields a total of

$$\begin{aligned}
T(m) \ &\leq \ (S-1) \cdot 2(s-1)(n - e_{s-1}) \\
&\quad +2(s-1)(m-n+1) - (S-1) \cdot 2(s-1)(n - e_{s-1}) \\
&= \ 2(s-1)(m-n+1)
\end{aligned}$$

operations as claimed. □

The cost estimate generalizes in a natural way the cost of $2n(m-n+1)$ for division with remainder of (dense) polynomials of degrees $m \geq n$ with monic divisor (see Section 2.4 of von zur Gathen and Gerhard, 2003).

Corollary 2.4. *Let $s = \sigma_q(n)$. Then*

- *two elements in $\mathbb{F}_{q^n}$ can be multiplied with at most*

$$\mathsf{M}(n) + 2(s-1)(n-1)$$

operations in $\mathbb{F}_q$,

- *an element can be raised to the* qth *power with at most*

$$2(s-1)(n-1) \quad \textit{or with} \quad 2\mathsf{M}(n)\log_2 q + 4(s-1)(n-1)\log_2 q$$

operations in $\mathbb{F}_q$.

Proof. The first claim follows from Theorem 2.3 by noting that the product of two representatives has degree $m \le 2(n-1)$, so that

$$2(s-1)(m-n+1) \;\le\; 2(s-1)(2n-2-n+1) = 2(s-1)(n-1).$$

There are two possibilities to evaluate a qth power. One may profit from the observation that the qth power of $g = \sum_{0\le i<n} g_i x^i$ is just $g^q = \sum_{0\le i<n} g_i x^{iq}$ with $\deg(g^q) = m \le q(n-1)$. This can be evaluated without operations in $\mathbb{F}_q$. A final division with remainder yields the first estimate. Repeated squaring for the qth power yields the second estimate. □

In the case $q = 2$ squaring can be done with cost at most $2(n-1)(s-1)$. We worked out a table of irreducible trinomials. For 5146 values of n in the range $2 \le n < 10{,}000$ there exists an irreducible trinomial of degree n in $\mathbb{F}_2[x]$, see also Zierler and Brillhart (1968). For the remaining 4852 values, there exist irreducible pentanomials. Swan (1962) discusses the number of irreducible factors for trinomials in $\mathbb{F}_2[x]$, and shows that $\sigma_2(n) \ge 5$ when n is a positive integer multiple of 8.

The factorization of trinomials over $\mathbb{F}_2$ is also discussed in Chapter 5 of Golomb (1967). Further tables of irreducible trinomials in $\mathbb{F}_2[x]$ are given in Zierler and Brillhart (1969), Zierler (1970), and Fredricksen and Wisniewski (1981), and some of large degree in Brent et al. (2003). See Loidreau (2000) and von zur Gathen (2003) for trinomials over $\mathbb{F}_3$. Interestingly, in the experiments of the latter paper trinomials turned out to be irreducible slightly more often than general polynomials. These computations and Tables 2.1 and 2.2 motivate the following conjecture.

Table 2.1. Irreducible sparse polynomials of degree n over prime fields $\mathbb{F}_q$ with $\sigma_q(n)$ nonzero coefficients.

Degree	Polynomial ring $\mathbb{F}_q[x]$			
n	$\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_5$	$\mathbb{F}_7$
1	$x+1$	$x+1$	$x+1$	$x+1$
2	x^2+x+1	x^2+1	x^2+2	x^2+1
3	x^3+x+1	x^3+2x+1	x^3+x+1	x^3+2
4	x^4+x+1	x^4+x+2	x^4+2	x^4+x+1
5	x^5+x^2+1	x^5+2x+1	x^5+4x+1	x^5+x+3
6	x^6+x+1	x^6+x+2	x^6+x+2	x^6+2
7	x^7+x+1	x^7+x^2+2	x^7+x+1	x^7+6x+1
8	$x^8+x^4+x^3+x+1$	x^8+x^2+2	x^8+2	x^8+x+3
9	x^9+x+1	x^9+x^4+2	x^9+x^4+4	x^9+2
10	$x^{10}+x^3+1$	$x^{10}+2x^2+1$	$x^{10}+4x^2+2$	$x^{10}+2x+3$

Table 2.2. Irreducible sparse polynomials of degree n over finite fields $\mathbb{F}_q$ with $\sigma_q(n)$ nonzero coefficients. We adjoin a root a of an irreducible polynomial over the prime field of $\mathbb{F}_q$ to construct $\mathbb{F}_q$. The chosen modulus is given in row 2.

Degree	Polynomial ring $\mathbb{F}_q[x]$ with composite q		
n	$\mathbb{F}_4 = \mathbb{F}_2[y]/(y^2+y+1)$	$\mathbb{F}_8 = \mathbb{F}_2[y]/(y^3+y+1)$	$\mathbb{F}_9 = \mathbb{F}_3[y]/(y^2+1)$
1	$x+1$	$x+1$	$x+1$
2	x^2+x+a	x^2+x+1	x^2+1+a
3	x^3+a	x^3+x+a	x^3+x+a
4	x^4+x^2+ax+1	x^4+x+1	$x^4+(1+a)$
5	x^5+x+a	x^5+x^2+1	$x^5+x+(1+a)$
6	x^6+x^3+a	x^6+x+a	$x^6+x^2+(1+a)$
7	x^7+x+1	x^7+a	$x^7+x+(1+a)$
8	x^8+x^3+x+a	$x^8+x^3+ax+(1+a)$	$x^8+(1+a)$
9	x^9+a	$x^9+x+(1+a)$	x^9+x^2+a
10	$x^{10}+x^5+a$	$x^{10}+x^3+1$	$x^{10}+x+(2+a)$

Conjecture 2.5. *For all $n, q \in \mathbb{N}_{\geq 2}$ with q a prime power, we have $\sigma_q(n) \leq 5$. If $q \geq 3$, then $\sigma_q(n) \leq 4$.*

We have $\sigma_3(n) = 4$ for the six values 49, 57, 65, 68, 75, and 98 of $n \leq 100$. A summary on results discussing the complete factorization of sparse polynomials over a prime field $\mathbb{F}_p$ is given in the book of Shparlinski (1999, Sections 3.2 and 3.3).

Corollary 2.6. *Let $\mathbb{F}_{2^n}$ be defined by an s-sparse polynomial f with $\sigma_2(n)$ nonzero entries. If* Conjecture 2.5 *is true, then two elements in $\mathbb{F}_{2^n}$ can be multiplied with at most $\mathsf{M}(n) + 8n - 8$ operations in $\mathbb{F}_2$. Squaring can be done with $8n - 8$ operations in $\mathbb{F}_2$. A power of an element in $\mathbb{F}_{2^n}$ can be computed with at most*

$$(\mathsf{M}(n) + 8n - 8)\frac{n}{\log n}(1 + o(1)) + 8n^2 - 8n$$

operations in $\mathbb{F}_2$.

Corollary 2.7. *Assume that* Conjecture 2.5 *is true and that $\mathsf{M}(n) \in \Omega(n \log n)$. Then an element of $\mathbb{F}_{2^n}$ can be raised to a power with $O(\mathsf{M}(n)(n/\log n)) \subseteq O(n^2 \log\log n)$ operations in $\mathbb{F}_2$.*

Sedimentary polynomials. Coppersmith's (1984) algorithm for solving the discrete logarithm problem in characteristic 2 uses t-sedimentary polynomials. His idea and further improvements of it are also discussed in Odlyzko (1985). He remarks on the existence of such polynomials: "Choose a primitive polynomial $P(x)$ of degree n, such that $P(x) = x^n + Q(x)$, where the degree of $Q(x)$ is smaller than $n^{2/3}$. (This should be possible; heuristically, for a given n, we expect the best possible $Q(x)$ to have degree about $\log_2 n$.)" This describes a t-sedimentary polynomial $f = x^n + h$ with $t = \deg h \ll n$. Sedimentarity is a special case of sparseness with $s \leq t + 2$. A fraction of about $1/n$ of all polynomials of degree n is irreducible; more detailed bounds are given in Hardy and Wright (1985). Thus heuristically one might hope for

Table 2.3. Irreducible sedimentary polynomials of degree n over prime fields $\mathbb{F}_q$. The sediment $h = f - x^n$ has degree $\tau_q(n)$.

Degree n	Polynomial ring $\mathbb{F}_q[x]$ $\mathbb{F}_2$	$\mathbb{F}_3$	$\mathbb{F}_5$	$\mathbb{F}_7$
1	$x+1$	$x+1$	$x+1$	$x+1$
2	x^2+x+1	x^2+1	x^2+2	x^2+1
3	x^3+x+1	x^3+2x+1	x^3+x+1	x^3+2
4	x^4+x+1	x^4+x+2	x^4+2	x^4+x+1
5	x^5+x^2+1	x^5+2x+1	x^5+4x+1	x^5+x+3
6	x^6+x+1	x^6+x+2	x^6+x+2	x^6+2
7	x^7+x+1	x^7+x^2+2	x^7+x+1	x^7+6x+1
8	$x^8+x^4+x^3+x+1$	x^8+x^2+2	x^8+2	x^8+x+3
9	x^9+x+1	$x^9+2x^3+x^2+1$	x^9+x^2+2x+3	x^9+2
10	$x^{10}+x^3+1$	$x^{10}+2x^2+1$	$x^{10}+x^2+x+3$	$x^{10}+2x+3$

$\tau_q(n) = \min\{\deg h\colon x^n + h \in \mathbb{F}_q[x]$ is irreducible$\}$ to be roughly $\log_q n$, since there are about n polynomials with degree up to $\log_q n$. Indeed, we found $\tau_2(n) \leq 2 + \lceil \log_2 n \rceil \ll n$ for all tested n (see Table 2.6, column 5, and Table 4.2, column 7). Gordon and McCurley (1992) found $\tau_2(n) \leq 11$ for all $n \leq 600$. The experiments of Gao and Panario (1997) and Gao et al. (1999) showed $\tau_2(n) \leq 3 + \log_2 n$ for $q = 2$ and all $n < 2000$. Our own calculations validate this bound on $\tau_2(n)$ for all $n \leq 5000$. The following conjecture is motivated by these experiments and Tables 2.3 and 2.4.

Conjecture 2.8. *For all $n, q \in \mathbb{N}_{>0}$, with $q \geq 2$ a prime power, we have $\tau_q(n) \leq 3 + \log_q n$.*

We have chosen the numerical parameters in our conjectures about sparse and sedimentary polynomials in the strongest form compatible with our experimental results, thus

Table 2.4. Irreducible sedimentary polynomials of degree n over finite fields $\mathbb{F}_q$. The sediment $h = f - x^n$ has degree $\tau_q(n)$. We adjoin a root a of an irreducible polynomial over the prime field of $\mathbb{F}_q$ to construct $\mathbb{F}_q$. The chosen modulus is given in row 2.

Degree n	Polynomial ring $\mathbb{F}_q[x]$ with composite q: $\mathbb{F}_4 = \mathbb{F}_2[y]/(y^2+y+1)$	$\mathbb{F}_8 = \mathbb{F}_2[y]/(y^3+y+1)$	$\mathbb{F}_9 = \mathbb{F}_3[y]/(y^2+1)$
1	$x+1$	$x+1$	$x+1$
2	x^2+x+a	x^2+x+1	$x^2+(1+a)$
3	x^3+a	x^3+x+a	x^3+x+a
4	x^4+x^2+ax+1	x^4+x+1	$x^4+(1+a)$
5	x^5+x+a	x^5+x^2+1	$x^5+x+(1+a)$
6	x^6+x^2+x+a	x^6+x+a	$x^6+x^2+(1+a)$
7	x^7+x+1	x^7+a	$x^7+x+(1+a)$
8	x^8+x^3+x+a	$x^8+x^3+ax+(1+a)$	$x^8+(1+a)$
9	x^9+a	$x^9+x+1+a$	x^9+x^2+a
10	$x^{10}+x^3+ax^2+(1+a)$	$x^{10}+x^2+ax+1$	$x^{10}+x+(2+a)$

facilitating their refutation—if incorrect. For practical purposes, it is quite sufficient for the conjectures to hold for "most" degrees, or with one more term in the irreducible polynomials.

We rewrite the results of the previous paragraph for sedimentary polynomials $f = x^n + h \in \mathbb{F}_q[x]$ with $h \neq 0$ and $t = \deg h < n$. This special kind of sparseness yields $s \leq t + 2$ and $e_{s-1} = t$. Algorithm 2.2 works well in the case of t-sedimentary polynomials. We can apply polynomial multiplication in Step 4 of Algorithm 2.2 to compute g_1. This yields the following result.

Theorem 2.9. *Let $f, g \in \mathbb{F}_q[x]$ with $\deg g = m$ and let $f = x^n + h$ be a sedimentary polynomial with $0 \leq t = \deg h \leq (n-1)/2$. Division with remainder of g by f can be performed with at most $((m-n+1)/(t+1)+1)\mathsf{M}(t+1)+2(m-n+1)+t$ operations in $\mathbb{F}_q$.*

Proof. We substitute Step 3 in Algorithm 2.2 by

3′. Set $k = \max\{n, m-t\} \geq n$. Write $g = u_1 x^k + w$ with $u_1, w \in \mathbb{F}_q[x]$ and $\deg w < k$.

By assumption we have $n - e_{s-1} - 1 = n - t - 1 \geq n - (n-1)/2 - 1 = (n-1)/2$ which yields correctness for this modified choice of k. The cost is determined by the computation of $g_1 = w - u_1 \cdot (f - x^n)x^{k-n}$ in Step 4 of Algorithm 2.2. Now $f - x^n = h$ is a polynomial of degree t and u_1 has degree $m - k$. If $k = m - t$, then $\deg u_1 = t$. Else we have $k = n \geq m - t$ and $m - k = m - n \leq m - (m - t) = t$. In both cases $u_1 \cdot h$ can be evaluated with at most $\mathsf{M}(t+1)$ operations in $\mathbb{F}_q$ in Step 4. The resulting polynomial has at most $t + m - k + 1$ many nonzero coefficients. It can be subtracted from w with at most this number of operations in $\mathbb{F}_q$.

Let $T(m)$ denote the number of operations to compute (u, v) with Algorithm 2.2. Then $T(m) = 0$ if $m < n$. Else the problem size for the recursive call is the degree of g_1. If $m - t \geq n$, then $\deg g_1 = \max\{\deg w, \deg u_1 h + k - n\} \leq \max\{m - (t+1), 2t + m - t - n\} = m - (t+1)$ since $2t \geq n - 1$ by assumption. Thus after a recursive call the problem size is decreased by $t + 1$. Let S be the number of recursive calls until the algorithm stops. Then $m - S \cdot (t+1) \leq n - 1 < m - (S-1) \cdot (t+1)$ which yields $S \leq \lceil (m-n+1)/(t+1) \rceil$. For $S - 1$ calls we have $k = m - t$ and thus $t + m - k + 1 = 2t + 1$ operations in $\mathbb{F}_q$ in Step 4. For the final call we have $k = n$ and $\deg u_1 = m - (S-1)(t+1) - n$ with m the input degree of the original call. Then Step 4 causes $t + (m - (S-1)(t+1) - n) + 1$ operations in $\mathbb{F}_q$. Thus we have a total of

$$\begin{aligned} T(m) &\leq S \cdot \mathsf{M}(t+1) + (S-1) \cdot (2t+1) + m - n + 1 - (S-1)(t+1) + t \\ &= S \cdot \mathsf{M}(t+1) + m - n + 1 + t \cdot (S-1) + t \\ &\leq \left\lceil \frac{m-n+1}{t+1} \right\rceil \cdot \mathsf{M}(t+1) + m - n + 1 + t \cdot \frac{m-n+1}{t+1} + t \\ &\leq \left\lceil \frac{m-n+1}{t+1} \right\rceil \cdot \mathsf{M}(t+1) + 2(m-n+1) + t \end{aligned}$$

operations in $\mathbb{F}_q$ as claimed. □

The qth power g^q of a polynomial $g \in \mathbb{F}_q[x]$ with $\deg g \le n-1$ has degree at most $m = q(n-1)$ and is computed without arithmetic operations. Using this to exponentiate in $\mathbb{F}_{q^n}$, we have the following analog of Corollary 2.4.

Corollary 2.10. *Let $f = x^n + h \in \mathbb{F}_q[x]$ be irreducible of degree n, and $t = \deg h \le (n-1)/2$. Then two elements in $\mathbb{F}_q[x]/(f)$ can be multiplied with at most*

$$\mathsf{M}(n) + \left(\frac{n-1}{t+1} + 1\right) \cdot \mathsf{M}(t+1) + 2(n-1) + t$$

*operations in $\mathbb{F}_q$, and an element can be raised to the q*th *power with at most*

$$\left(\frac{(q-1)(n-1)}{t+1} + 1\right) \mathsf{M}(t+1) + 2(q-1)(n-1) + t$$

operations in $\mathbb{F}_q$.

Finally we formulate this result for the case $q = 2$. We assume fast multiplication with $\mathsf{M}(t+1) \in O(t \log t \log\log t)$ for the asymptotic estimate. For implementations the sedimentary part h is of very small degree compared with $\deg f = x^n + h$, and then classical multiplication with $\mathsf{M}(t+1) = 2(t+1)^2$ would be used.

Corollary 2.11. *Let $\mathbb{F}_{2^n}$ be defined by a sedimentary polynomial $f = x^n + h$ with* $\deg h = \tau_2(n)$, *and assume Conjecture* 2.8 *to be true.*

- *Two elements in $\mathbb{F}_{2^n}$ can be multiplied with at most*

$$\mathsf{M}(n) + \left(\frac{n}{\log n}\mathsf{M}(\log n)\right) \in O(n \log n \log\log n)$$

  operations in $\mathbb{F}_2$.
- *Squaring can be done with*

$$O\left(\frac{n}{\log n}\mathsf{M}(\log n)\right) \in O(n \log\log n \log\log\log n)$$

  operations in $\mathbb{F}_2$.
- *A power of an element in $\mathbb{F}_{2^n}$ can be computed with at most*

$$\mathsf{M}(n)\frac{n}{\log n}(1 + o(1)) + O\left(\frac{n^2}{\log n}\mathsf{M}(\log n)\right) \in O(n^2 \log\log n \log\log\log n)$$

  operations in $\mathbb{F}_2$.

Experimental results. We have implemented in `C++` some exponentiation algorithms over $BbbF_{2^n}$, using the software library BiPolar written by Jürgen Gerhard for fast polynomial arithmetic over $\mathbb{F}_2$; for details see Section 9.7 of von zur Gathen and Gerhard (2003). Three different polynomial multiplication algorithms are available: classical multiplication with $\mathsf{M}(n) \in O(n^2)$ is used for $n < 576$. The subquadratic algorithm

Table 2.5. Multiplication and squaring in $\mathbb{F}_2[x]/(f)$ for dense and sparse f. The running time (in CPU milliseconds on a SUN Sparc ULTRA-IIi) is the average of 1000 experiments for each value of n.

	Random f		Sedimentary f			Random f		Sedimentary f	
n	Mult.	Square	Mult.	Square	n	Mult.	Square	Mult.	Square
209	0.1	0.1	0.1	0.0	5199	13.9	9.1	5.5	0.3
398	0.3	0.2	0.1	0.0	5399	14.6	9.7	5.7	0.3
606	0.5	0.3	0.3	0.0	5598	15.1	9.8	5.9	0.3
803	0.7	0.4	0.4	0.1	5812	15.8	10.5	6.2	0.3
1018	0.9	0.7	0.4	0.1	6005	16.3	11.0	6.3	0.4
1199	1.4	0.9	0.6	0.1	6202	17.1	11.0	6.6	0.3
1401	1.7	1.2	0.7	0.1	6396	18.0	12.0	6.9	0.3
1601	2.1	1.4	0.9	0.1	6614	18.5	12.3	7.2	0.4
1791	2.5	1.7	1.0	0.1	6802	19.1	11.9	7.5	0.4
1996	3.0	2.2	1.1	0.1	7005	19.5	12.9	7.6	0.4
2212	3.8	2.6	1.5	0.2	7205	20.2	13.4	8.0	0.6
2406	4.4	3.0	1.7	0.2	7410	20.7	13.9	8.0	0.4
2613	5.1	3.5	1.9	0.2	7602	21.2	14.1	8.2	0.4
2802	5.6	4.0	2.0	0.2	7803	21.5	14.4	8.3	0.5
3005	6.3	4.5	2.2	0.2	8003	22.0	14.7	8.6	0.7
3202	7.1	5.1	2.5	0.2	8218	23.9	15.9	9.1	0.4
3401	7.8	5.7	2.7	0.3	8411	29.5	17.8	10.4	0.5
3603	8.6	6.4	2.7	0.2	8601	30.1	20.1	11.2	0.5
3802	9.3	7.0	2.8	0.2	8802	32.0	21.5	12.2	0.6
4002	10.1	7.8	3.0	0.3	9006	33.3	22.3	12.5	0.5
4211	9.2	6.1	3.6	0.3	9202	34.6	23.2	12.8	0.5
4401	10.4	7.0	4.1	0.3	9396	36.6	24.0	13.6	0.5
4602	11.3	7.6	4.4	0.3	9603	37.8	24.9	15.5	0.9
4806	13.0	9.1	5.0	0.3	9802	39.0	26.0	15.2	0.7
5002	12.9	8.6	5.2	0.3	9998	39.8	25.9	14.9	0.5

of Karatsuba—described in Karatsuba and Ofman (1962)—with $\mathsf{M}(n) \in O(n^{\log_2 3})$ is applied for $576 \le n < 35{,}840$. For larger n the library chooses a fast multiplication routine based on Cantor (1989) which is nearly linear: $\mathsf{M}(n) \in O(n(\log n)^2)$.

The library also contains an implementation of the modular composition algorithm of Brent and Kung (1978), using classical matrix multiplication. The sedimentary division with remainder has been implemented by Olaf Müller for $q = 2$; we added division by trinomials to the arithmetic package of BiPolAr. These special versions are significantly faster than the implementation for general divisors in our range of inputs as documented in Table 2.5. We did not experiment with pentanomials.

Table 2.6 shows experimental results on a SUN Sparc ULTRA-IIi rated at 269.5 MHz for the three algorithms discussed for $\mathbb{F}_q[x]/(f)$ in this section. We choose trinomials as well as sedimentary polynomials for the implementations of Algorithm 2.2. We use a first test *Series Linear* with $n \approx 200i$ and $1 \le i \le 50$. This includes the range for cryptographic applications nowadays. Each entry is the average for 100 pairs of base and exponent chosen at random. The irreducible polynomials defining $\mathbb{F}_{2^n}$ are chosen at random for the columns labeled 2.1 and "polynomial representation of the Frobenius," which we now call the Frobenius method for short.

Table 2.6. Comparison of exponentiation for different polynomial basis representations of $\mathbb{F}_{2^n}$. The running time (in CPU seconds on a SUN SparcULTRA-IIi) is the average of 100 experiments for each value n. The chosen n (column 6) for trinomials differs from the one in column 1 if no irreducible trinomial of degree n (column 1) over $\mathbb{F}_2$ exists.

			Corollary 2.11 $f = x^n + h$			Corollary 2.6 $f = x^n + x^k + 1$	
n	Fact 2.1 rand. f time	Poly. rep. Frobenius time	Time	deg h	n	Time	k
209	0.02	0.03	0.01	5	209	0.01	6
398	0.09	0.15	0.03	7	399	0.02	26
606	0.25	0.37	0.07	9	606	0.06	165
803	0.51	0.63	0.12	8	804	0.09	75
1018	0.91	1.01	0.15	10	1020	0.12	135
1199	1.47	1.92	0.24	11	1199	0.21	114
1401	2.18	2.62	0.33	11	1401	0.28	92
1601	3.16	3.57	0.51	11	1601	0.39	548
1791	4.22	4.46	0.53	12	1791	0.46	190
1996	5.64	5.60	0.62	9	1996	0.54	307
2212	7.61	7.52	0.97	11	2212	0.78	423
2406	9.62	9.25	1.17	8	2407	0.95	91
2613	11.88	11.39	1.28	11	2614	1.13	553
2802	14.32	13.07	1.44	9	2801	1.29	279
3005	17.28	16.10	1.65	9	3004	1.46	351
3202	20.84	18.93	2.36	9	3201	1.71	674
3401	24.48	22.17	2.40	11	3401	1.94	531
3603	28.44	24.81	2.50	10	3604	2.09	637
3802	33.01	28.06	2.57	13	3801	2.30	112
4002	38.28	31.57	3.18	8	4001	2.45	137
4211	38.17	40.23	3.47	12	4212	3.05	243
4401	46.87	43.21	4.34	12	4401	3.89	394
4602	52.36	46.88	4.72	14	4602	4.30	67
4806	59.14	52.50	5.76	12	4806	4.90	2349
5002	65.02	56.39	6.18	11	5001	5.27	637
5199	70.37	64.00	6.36	12	5199	5.74	1546
5399	78.12	67.57	6.84	9	5399	6.24	485
5598	81.96	71.43	7.26	9	5598	6.62	101
5812	90.40	77.43	7.74	11	5812	7.17	295
6005	138.35	84.11	8.20	12	6006	7.57	1025
6202	101.60	86.94	8.76	12	6202	8.00	867
6396	109.68	92.88	9.39	12	6396	8.61	91
6614	116.60	98.53	9.77	12	6614	8.96	2105
6802	118.99	100.43	10.36	11	6801	9.52	140
7005	129.63	106.28	10.95	13	7004	9.89	291
7205	137.54	114.91	12.74	14	7204	10.43	1695
7410	147.29	120.37	12.02	10	7410	10.92	2179
7602	151.38	124.68	12.38	10	7602	11.53	555
7803	158.21	129.90	13.36	12	7802	11.82	2103
8003	165.40	134.23	15.59	8	8004	12.44	3087
8218	180.72	152.07	14.69	12	8218	13.39	1443
8411	207.55	176.35	16.87	12	8412	15.33	1049
8601	228.47	190.43	18.30	7	8601	17.21	734
8802	254.54	210.35	21.66	14	8802	18.30	2139
9006	270.58	222.12	21.79	9	9006	19.25	1477
9202	287.79	231.27	21.93	12	9202	20.04	211
9396	309.67	252.12	23.33	13	9396	22.14	369
9603	328.73	267.77	27.20	12	9601	26.15	963
9802	348.95	275.81	28.06	12	9801	24.49	284
9998	358.00	281.96	27.51	13	9998	24.82	4013

For random f we have a dominant term of at most $3n\mathsf{M}(n)$ in Fact 2.1 and of $O(\mathsf{M}(n)(n/\log n))$ in the Frobenius method. Indeed, the advantage of the Frobenius method is shown in the comparison of columns 2 and 3 in Table 2.6. The implementation of the Frobenius method is superior by a factor of roughly $\frac{1}{10}\log_2 n$ compared with straightforward exponentiation with Brauer's addition chain. For sedimentary polynomials and trinomials of degree $n < 10{,}000$, the dominating cost is $\mathsf{M}(n)(n/\log n)$ by Corollaries 2.11 and 2.6, respectively, since $\mathsf{M}(n) \in O(n^{\log_2 3})$. This is asymptotically the same as for modular composition. Indeed, column 3 grows asymptotically at the same rate as columns 4 and 7, but with a factor of about 10 for sedimentary polynomials and about 11 for trinomials, respectively. This constant factor makes sparse polynomials the clear winner.

3. Normal Basis Representation

If $\alpha \in \mathbb{F}_{q^n}$ is such that its conjugates $\alpha, \alpha^q, \alpha^{q^2}, \ldots, \alpha^{q^{n-1}}$ form a vector space basis for $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$, then α is *normal* over $\mathbb{F}_q$, and $\mathcal{N} = (\alpha_0, \ldots, \alpha_{n-1})$ is a *normal basis*. A normal basis exists for all finite fields (seeTheorem 2.35 of Lidl and Niederreiter, 1983). There are different ways to multiply in a normal basis representation. The efficient ones only work for special choices of α.

Classical arithmetic. Let $\mathcal{N} = (\alpha, \ldots, \alpha^{q^{n-1}})$ be a normal basis of $\mathbb{F}_{q^n}$ over $\mathbb{F}_q$. Then any $\beta \in \mathbb{F}_{q^n}$ can be given by its *normal basis representation* $\beta = \sum_{0\le i<n} b_i\alpha_i$, where $b_0, \ldots, b_{n-1} \in \mathbb{F}_q$. Let $\sigma: \mathbb{F}_{q^n} \to \mathbb{F}_{q^n}$ with $\sigma(\beta) = \beta^q$ be the *Frobenius automorphism*. It is a linear operator on $\mathbb{F}_{q^n}$ as an $\mathbb{F}_q$-vector space. We have $\beta^q = \sigma(\beta) = \sigma(\sum_{0\le i<n} b_i\alpha_i) = \sum_{0\le i<n} b_i\sigma(\alpha_i) = \sum_{0\le i<n} b_i\alpha_{i+1} = \sum_{0\le i<n} b_{i-1}\alpha_i$, with index arithmetic modulo n. Hence raising to the qth power is just a cyclic shift of the coordinates and therefore essentially free. With an appropriate *q-addition chain*, a power in $\mathbb{F}_{q^n}$ can be computed with $O(n/\log_q n)$ operations in $\mathbb{F}_{q^n}$ (von zur Gathen, 1991).

Multiplication is more difficult and expensive. We define a *multiplication matrix* $T_{\mathcal{N}} = (t_{i,j})_{0\le i,j<n}$ such that $\alpha_i\alpha_j = \sum_{0\le h<n} t_{i-h,h-j}\alpha_h$ for all $0 \le i, j < n$. Details of the corresponding *Massey–Omura multiplier*—designed for hardware applications—are given in Chapter 5 of Menezes et al. (1993). We call the number of nonzero entries in $T_{\mathcal{N}}$ the *density* $d_{\mathcal{N}}$. Then two elements of $\mathbb{F}_{q^n}$ can be multiplied with at most $2nd_{\mathcal{N}}$ multiplications in $\mathbb{F}_q$.

Obviously $d_{\mathcal{N}} \le n^2$. Mullin et al. (1989) prove $2n - 1 \le d_{\mathcal{N}}$ as a lower bound on $d_{\mathcal{N}}$. They call a normal basis $\mathcal{N}$ with $d_{\mathcal{N}} = 2n - 1$ *optimal*.

Fact 3.1. *Let $\mathbb{F}_{q^n}$ be given by a normal basis representation. We can compute any power in $\mathbb{F}_{q^n}$ with $2d_{\mathcal{N}}(n^2/\log n)(1 + o(1)) \in O(n^4/\log n)$ operations in $\mathbb{F}_q$. If the normal basis is optimal then we have at most $4(n^3/\log n)(1 + o(1))$ operations in $\mathbb{F}_q$.*

An optimal normal basis does not exist for all n and q, but seems to exist for a reasonably dense set of values of n, e.g., for 23% of all $n \le 1200$ if $q = 2$ (Mullin et al.,

Table 3.1. Percentage of fields $\mathbb{F}_{q^n}$ with $n \leq 10{,}000$ for which there exists an optimal normal basis over $\mathbb{F}_q$.

q	2	3	5	7	11	13	17	19
%	17.07*	4.92	4.92	4.65	4.43	4.57	4.50	4.72

* We have two different types of optimal normal bases only for $q = 2$: the first one appears in 4.70%, the second one exists in 12.37% of the field extensions over $\mathbb{F}_2$.

1989). The percentage of fields $\mathbb{F}_{q^n}$ for which optimal normal bases do exist for some small primes q and $n \leq 10000$ is given in Table 3.1.

Gauss periods

Definition 3.2. Let $n, k \in \mathbb{N}_{\geq 1}$ such that $nk + 1$ is prime. Let $\mathcal{K} \subseteq \mathbb{Z}_{nk+1}^{\times}$ be the unique subgroup of $\mathbb{Z}_{nk+1}^{\times}$ of order k, and let ξ be a primitive $(nk + 1)$st root of unity in $\mathbb{F}_{q^{nk}}$. Then $\alpha = \sum_{a \in \mathcal{K}} \xi^a$ is called a *Gauss period of type* (n, k) over $\mathbb{F}_q$.

Fact 3.3 (Wassermann, 1990, 1993). *Let α be a Gauss period of type (n, k) and let $\mathcal{K}$ be the uniquely determined subgroup of $\mathbb{Z}_{nk+1}^{\times}$ of order k. Then α is normal in $\mathbb{F}_{q^n}$ if and only if q and $\mathcal{K}$ together generate $\mathbb{Z}_{nk+1}^{\times}$.*

Mullin et al. (1989) showed that Gauss periods of type $(n, 1)$ and $(n, 2)$ generate optimal normal bases. Gao and Lenstra (1992) proved that the constructions presented by Mullin et al. (1989) cover all optimal normal bases.

Normal bases with fast polynomial multiplication. Gao et al. (2000) have combined fast polynomial multiplication and normal bases if the normal element α is generated by a Gauss period.

Fact 3.4 (Gao et al., 2000). *Let $\alpha \in \mathbb{F}_{q^n}$ be a normal Gauss period of type (n, k). Then two elements in $\mathbb{F}_{q^n}$ given in the normal basis representation generated by α can be multiplied with $\mathsf{M}(kn) + (2k + 1)n - 2$ operations in $\mathbb{F}_q$.*

Corollary 3.5. *Let $\alpha \in \mathbb{F}_{q^n}$ be a normal Gauss period of type (n, k) and let $\mathbb{F}_{q^n}$ be represented in the normal basis $\mathcal{N} = (\alpha, \ldots, \alpha^{q^{n-1}})$. We can compute a power in $\mathbb{F}_{q^n}$ with $(n/\log n)(\mathsf{M}(kn) + (2k + 1)n - 2)(1 + o(1)) \in O(kn^2 \log\log(kn)(1 + \log k))$ operations in $\mathbb{F}_q$. If α is optimal, then we have $O(n^2 \log\log n)$ operations in $\mathbb{F}_q$.*

Experiments. We concentrate on $\mathbb{F}_{2^n}$ using BIPOLAR again. We represent 32 coefficients in one machine word when implementing the normal basis representation of $\mathbb{F}_{2^n}$ in C++. All experiments are confined to optimal normal bases.

We implemented the classical normal basis multiplication (Fact 3.1) using the multiplication matrix $T_{\mathcal{N}}$. Details on this *Massey–Omura multiplier* are given in the patent of Omura and Massey (1986). Our implementation profits from the distribution of nonzero entries in $T_{\mathcal{N}}$ in the case $k = 1$. Thus the multiplication time is reduced to roughly $\frac{3}{5}$ compared with the times for $k = 2$.

Table 3.2. Comparison of the two exponentiation algorithms for normal basis representation. The running time (in CPU seconds on a SUN Sparc ULTRA-IIi) is the average of 100 experiments for each value of n. All normal elements are generated by Gauss periods of type (n, k) with $k \in \{1, 2\}$. Such normal elements are called *optimal*.

n	k	Fact 3.1 Massey–Omura	Corollary 3.5 Gauss period	n	k	Fact 3.1 Massey–Omura	Corollary 3.5 Gauss period
209	2	0.11	0.01	5199	2	1061.87	15.28
398	2	0.64	0.04	5399	2	1183.07	16.65
606	2	2.19	0.12	5598	2	1307.12	17.80
803	2	4.56	0.20	5812	1	947.32	7.16
1018	1	6.07	0.12	6005	2	1678.64	20.02
1199	2	14.73	0.50	6202	1	1142.87	7.95
1401	2	23.03	0.68	6396	1	1227.81	8.36
1601	2	34.11	0.98	6614	2	2080.97	25.27
1791	2	47.16	1.15	6802	1	1485.51	9.68
1996	1	41.21	0.56	7005	2	2644.03	27.42
2212	1	55.03	0.81	7205	2	2680.38	28.58
2406	2	108.17	3.12	7410	1	1930.91	11.08
2613	2	138.94	3.00	7602	1	1958.51	11.42
2802	1	107.77	1.30	7803	2	3344.95	33.28
3005	2	215.04	4.01	8003	2	3689.48	34.62
3202	1	159.40	1.72	8218	1		13.85
3401	2	314.63	5.10	8411	2		43.67
3603	2	375.25	5.54	8601	2		47.98
3802	1	258.59	2.38	8802	1		18.42
4002	1	303.90	2.52	9006	2		54.55
4211	2	601.99	8.34	9202	1		21.02
4401	2	678.34	10.33	9396	1		22.64
4602	1	471.36	4.37	9603	2		64.36
4806	2	845.78	13.00	9802	1		24.95
5002	1	580.68	5.38	9998	2		69.77

For Gauss periods we have implemented the polynomial multiplication of Theorem 3.4. The results are listed in Table 3.2. In columns 4 and 8 of Table 3.2, we see that for neighboring values of n, $k = 2$ leads to a constant factor times the cost for $k = 1$. Since $n < 10{,}000$ the preferred multiplication algorithm is the one of Karatsuba. It has time $\mathsf{M}(n) = O(n^{\log_2 3})$ which yields $\mathsf{M}(2n) = 3\mathsf{M}(n)$ in theory. Our implementation shows a factor of 2.66 on average.

We select our n such that an optimal normal basis for $\mathbb{F}_{2^n}$ over $\mathbb{F}_2$ exists. Thus $k \in \{1, 2\}$ in columns 2 and 6 of Table 3.2. We again choose 100 values for each n at random in test *Series Linear*. It shows the same significant difference between both algorithms in theory and by experiments, as is visible in Fig. 1. In theory the classical multiplication is $O(n^3/\log n)$. For $kn < 20{,}000$ BiPoLAR uses the multiplication algorithm of Karatsuba and Ofman (1962) with $\mathsf{M}(kn) \leq 27(kn)^{\log_2 3}$. For the Gauss periods involved this yields $O(n^{2.59}/\log n)$.

4. Final Comparison

We summarize the theoretical results of Section 3 in Table 4.1. Using fast multiplication with $\mathsf{M}(n) \in O(n^2 \log n \log\log n)$, the asymptotic behavior is roughly quadratic for

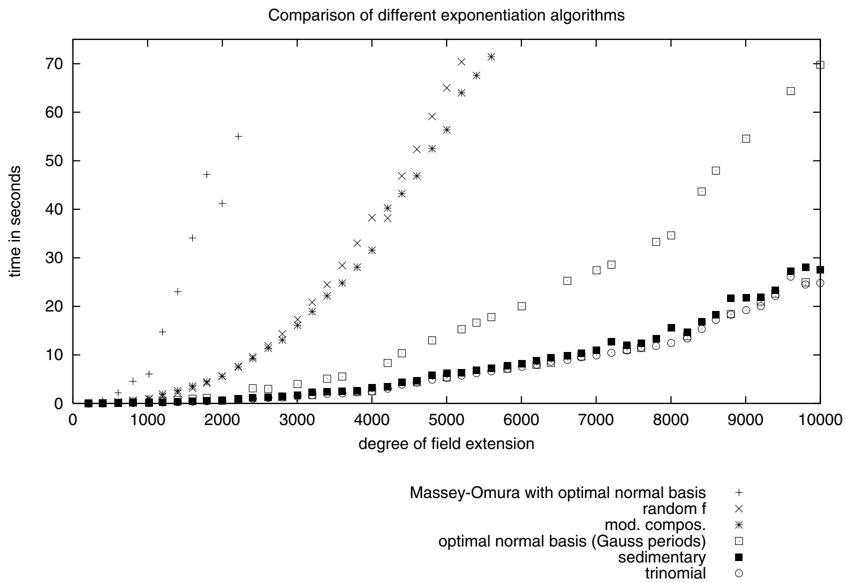

Fig. 1. Graphical representation of the results for test *Series Linear* as given in Tables 2.6 and 3.2.

all representations except the matrix-based optimal normal basis multiplication à la Omura and Massey (1986) (Fact 3.1). We do not have a clear winner, but the latter is a loser.

Table 4.2 gives the running times of a second test *Series Exp*. Here we choose n near 2^i for $10 \leq i \leq 16$, plus some intermediate values for n, driven by the condition that

Table 4.1. The weights and number of steps for exponentiation using different representations of $\mathbb{F}_{2^n}$.

Representation	Multiplication	Squaring	Operations in $\mathbb{F}_2$
Polynomial basis			
Random f (Fact 2.1)	$3\mathsf{M}(n) + O(n)$	$2\mathsf{M}(n) + O(n)$	$O(n^2 \log n \log\log n)$
Mod. comp.	$3\mathsf{M}(n) + O(n)$	$O(n^{1.688})$	
Sparse f^* (Cor. 2.6)	$\mathsf{M}(n) + O(n)$	$8n - 8$	$O(n^2 \log\log n)$
Sedimentary $f^\dagger$ (Cor. 2.11)	$\mathsf{M}(n) + O((n/\log n)\mathsf{M}(\log n))$	$O((n/\log n)\mathsf{M}(\log n))$	$O(n^2 \log\log n \log\log\log n)$
Normal basis			
Matrix $T_{\mathcal{N}}{}^\ddagger$ (Fact 3.1)	$4n^2 - 2n$	0	$O(n^3/\log n)$
Gauss periods§ (Cor. 3.5)	$\mathsf{M}(kn) + O(kn)$	0	$O(kn^2 \log k \log\log kn)$

* Assumes $\sigma_2(n) \leq 5$.
† Assumes $\tau_2(n) \in O(\log n)$.
‡ For optimal normal bases.
§ Only for some n.

Table 4.2. Exponentiation in $\mathbb{F}_{2^n}$ for test *Series Exp*. Bases and exponents are chosen randomly. The times are averages for ten random experiments. Again the choice of n for trinomials (column 8) differs if no irreducible trinomial of degree n (as given in column 1) exists over $\mathbb{F}_2$.

	Normal basis		Polynomial basis						
	Corollary 3.5 Gauss		Fact 2.1 rand. f	Poly. rep. Frobenius	Corollary 2.11 $f = x^n + h$			Corollary 2.6 $f = x^n + x^k + 1$	
n	k	t/sec	t/sec	t/sec	t/sec	deg h	n	t/sec	k
1,018	1	0.12	0.88	0.91	0.16	10	1,020	0.01	135
1,034	2	0.31	0.97	1.05	0.18	10	1,034	0.01	75
2,140	1	0.71	6.63	6.16	0.81	12	2,140	0.08	283
2,141	2	1.79	6.85	5.99	0.81	6	2,142	0.07	69
4,211	2	8.23	38.17	40.23	3.47	12	4,212	0.31	243
4,218	1	3.13	37.01	30.61	3.49	14	4,218	0.31	287
8,292	1	14.57	195.10	137.00	17.72	12	8,292	1.47	637
8,325	2	41.38	199.71	142.54	17.85	13	8,324	1.54	1,149
16,679	2	269.42	1,159.00	728.72	90.97	14	16,679	8.10	6,692
16,692	1	76.61	1,152.56	712.03	84.79	9	16,692	8.11	2,115
23,898	1	188.86	3,061.21	1,717.91	202.09	11	23,898	19.36	3,459
23,903	2	490.47	3,036.22	1,789.66	205.00	14	23,903	19.39	2,891
32,075	2	901.09	5,425.19	3,031.90	385.18	12			
32,076	1	339.49	5,408.11	2,930.06	383.56	15	32,076	34.51	1,825
43,371	2	1,756.39	1,1211.40	5,784.43	915.43	16	43,372	85.64	11,097
43,396	1	830.10	11,203.90	5,761.00	921.28	17	43,396	85.91	10,755
51,251	2	2,403.48	13,587.40	7,003.02	1,207.40	14	51,252	119.53	3,887
51,282	1	1,131.75	13,591.00	6,983.55	1,203.71	10	51,282	119.54	2,667
61,709	2	3,315.68	16,751.10	8,687.20	1,750.20	17	61,710	161.14	173
61,716	1	1,545.99	16,946.60	8,621.95	1,673.85	14	61,716	161.66	27,507

an optimal normal basis exists. Each entry is the average time for ten random choices of the base $\beta \in \mathbb{F}_{2^n}$ and exponent $e \in \mathbb{N}_{\geq 1}$.

We omit the algorithm of Omura and Massey (1986) since it is much too slow. The basic result is that sparse polynomials, both trinomials and sedimentary ones, are best to build $\mathbb{F}_{2^n}$. Gauss periods of type $(n, 1)$ are a good alternative but they exist only for fairly few values of n.

Ning and Yin (2001) study algorithms for normal basis multiplication and give timings for fields with up to 575 bits. They mention polynomial bases, but do not compare their results with that approach.

5. Conclusion

We have considered two ways of improving exponentiation algorithms in finite fields: reducing the number of operations in $\mathbb{F}_{q^n}$ and speeding up each operation. Both aspects are presented in theory as well as by implementation.

Our experiments show that it is worth while to work on a trade-off for the cost of qth powers and multiplications. Speeding up only one operation is not sufficient to achieve fast exponentiation, as shown by the high cost for multiplication in software implementations of the Omura and Massey (1986) algorithm. Of course, it was originally designed for hardware, and it was only recently found how to use Karatsuba's method efficiently in hardware (Grabbe et al., 2003). Our results—in theory as well as by experiment—suggest choosing as the data structure representing the finite field $\mathbb{F}_{q^n}$ either sparse irreducible polynomials or normal bases generated by optimal Gauss periods of type $(n, 1)$ over $\mathbb{F}_q$.

Algorithms that benefit from a special structure of the q-ary representation of the exponent—which occurs, e.g., in inversion and primitivity testing—are discussed in von zur Gathen and Nöcker (2003).

One question is left open in this paper: do there exist irreducible sparse polynomials as claimed in Conjectures 2.5 and 2.8?

Acknowledgements

The authors thank Jürgen Gerhard for helpful advice about library BIPOLAR, Olaf Müller for implementing sedimentary arithmetic in BIPOLAR, and the referees for useful remarks. Some of the results in this paper were reported in von zur Gathen and Nöcker (1997).

References

Alfred V. Aho, John E. Hopcroft and Jeffrey D. Ullman (1974). *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA.

A. Brauer (1939). On addition chains. *Bulletin of the American Mathematical Society* **45**, 736–739.

R. P. Brent and H. T. Kung (1978). Fast algorithms for manipulating formal power series. *Journal of the ACM* **25**(4), 581–595.

Richard P. Brent, Samuli Larvala and Paul Zimmermann (2003). A fast algorithm for testing reducibility of trinomials mod 2 and some new primitive trinomials of degree 3021377. *Mathematics of Computation* **72**, 1443–1452.

David G. Cantor (1989). On arithmetical algorithms over finite fields. *Journal of Combinatorial Theory, Series A* **50**, 285–300.

D. Coppersmith (1984). Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions on Information Theory* **IT-30**(4), 587–594.

Whitfield Diffie and Martin E. Hellman (1976). New directions in cryptography. *IEEE Transactions on Information Theory* **IT-22**(6), 644–654.

T. ElGamal (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* **IT-31**(4), 469–472.

H. Fredricksen and R. Wisniewski (1981). On trinomials $x^n + x^2 + 1$ and $x^{8l\pm 3} + x^k + 1$ irreducible over $GF(2)$. *Information and Control* **50**, 58–63.

S. Gao and H. W. Lenstra, Jr. (1992). Optimal normal bases. *Designs, Codes, and Cryptography* **2**, 315–323.

Shuhong Gao and Daniel Panario (1997). Tests and constructions of irreducible polynomials over finite fields. In *Foundations of Computational Mathematics* (Felipe Cucker and Michael Shub, editors), pp. 346–361. Springer-Verlag, New York.

Shuhong Gao, Jason Howell and Daniel Panario (1999). Irreducible polynomials of given forms. *Contemporary Mathematics* **225**, 43–54.

Shuhong Gao, Joachim von zur Gathen, Daniel Panario and Victor Shoup (2000). Algorithms for exponentiation in finite fields. *Journal of Symbolic Computation* **29**(6), 879–889. URL `http://dx.doi.org/10.1006/jsco.1999.0309`.

Joachim von zur Gathen (1991). Efficient and optimal exponentiation in finite fields. *Computational Complexity* **1**, 360–394.

Joachim von zur Gathen (2003). Irreducible trinomials over finite fields. *Mathematics of Computation* **72**, 1987–2000.

Joachim von zur Gathen and Jürgen Gerhard (2002). Polynomial factorization over $\mathbb{F}_2$. *Mathematics of Computation* **71**(240), 1677–1698. URL http://www-math.upb.de/∼aggathen/Publications/gatger02.pdf.

Joachim von zur Gathen and Jürgen Gerhard (2003). *Modern Computer Algebra*, 2nd edition. Cambridge University Press, Cambridge. ISBN 0-521-82646-2. URL http://www-math.upb.de/∼aggathen/mca/.

Joachim von zur Gathen and Michael Nöcker (1997). Exponentiation in finite fields: theory and practice. In *Applied Algebra*, *Algebraic Algorithms and Error-Correcting Codes*: *AAECC*-12, Toulouse, France (Teo Mora and Harold Mattson, editors), pp. 88–113. Number 1255 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISSN 0302-9743.

Joachim von zur Gathen and Michael Nöcker (2003). Computing special powers in finite fields. *Mathematics of Computation* **73**, 1499–1523.

Joachim von zur Gathen and Victor Shoup (1992). Computing Frobenius maps and factoring polynomials. *Computational Complexity* **2**, 187–224.

Solomon W. Golomb (1967). *Shift Register Sequences*. Holden-Day Series in Information Systems. Holden-Day, San Francisco, CA. With portions co-authored by Lloyd R. Welch, Richard M. Goldstein, and Alfred W. Hales.

Daniel M. Gordon and Kevin S. McCurley (1992). Massively parallel computation of discrete logarithms. In *Advances in Cryptology*: *Proceedings of CRYPTO ’*92, Santa Barbara, CA (Ernest F. Brickell, editor), pp. 312–323. Number 740 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISSN 0302-9743.

C. Grabbe, M. Bednara, J. Shokrollahi, J. Teich and J. von zur Gathen (2003). FPGA designs of parallel high performance $GF(2^{233})$ multipliers. In *Proceedings of the IEEE International Symposium on Circuits and Systems* (*ISCAS*-03), volume II, pp. 268–271.

G. H. Hardy and E. M. Wright (1985). *An introduction to the theory of numbers*. Clarendon Press, Oxford, 5th edition. First edition 1938.

A. Karatsuba and Yu. Ofman (1962). Умножение многозначных чисел на автоматах. Доклады Академий Наук СССР **145**, 293–294. A. Karatsuba and Yu. Ofman (1963). Multiplication of multidigit numbers on automata. *Soviet Physics—Doklady* **7**, 595–596.

Donald E. Knuth (1998). *The Art of Computer Programming*, *vol*. 2, *Seminumerical Algorithms*, 3rd edition. Addison-Wesley, Reading, MA.

Rudolf Lidl and Harald Niederreiter (1983). *Finite Fields*. Number 20 of Encyclopedia of Mathematics and its Applications. Addison-Wesley, Reading, MA.

Pierre Loidreau (2000). On the Factorization of Trinomials over $\mathbb{F}_3$. Technical Report 3918, Institut national de recherche en informatique et en automatique (INRIA). URL http://www.inria.fr/RRRT/RR-3918.html.

Alfred J. Menezes, Ian F. Blake, XuHong Gao, Ronald C. Mullin, Scott A. Vanstone and Tomik Yaghoobian (1993). *Applications of Finite Fields*. Kluwer, Norwell MA.

R. C. Mullin, I. M. Onyszchuk, S. A. Vanstone and R. M. Wilson (1989). Optimal normal bases in GF(p^n). *Discrete Applied Mathematics* **22**, 149–161.

Peng Ning and Yiqun Lisa Yin (2001). Efficient software implementation for finite field multiplication in normal basis. In *Proceedings of the Third International Conference on Information and Communications Security* 2001, Xian, China (Sihan Qing, Tatsuaki Okamoto and Jianying Zhou, editors), pp. 177–188. Number 2229 of Lecture Notes in Computer Science. Springer-Verlag, Berlin. ISBN 3-540-42880-1. ISSN 0302-9743. URL http://link.springer.de/link/service/series/0558/bibs/2229/22290177.htm.

A. Odlyzko (1985). Discrete logarithms and their cryptographic significance. In *Advances in Cryptology*: *Proceedings of EUROCRYPT* 1984, Paris, France, pp. 224–314. Springer-Verlag, New York.

Jimmy K. Omura and James L. Massey (1986). Computational method and apparatus for finite field arithmetic. United States Patent 4,587,627. URL http://www.patents.ibm.com/details?pn=US04587627__. Date of Patent: May 6, 1986.

A. Schönhage (1977). Schnelle Multiplikation von Polynomen über Körpern der Charakteristik 2. *Acta Informatica* **7**, 395–398.

A. Schönhage and V. Strassen (1971). Schnelle Multiplikation großer Zahlen. *Computing* **7**, 281–292.

Igor E. Shparlinski (1999). *Finite Fields*: *Theory and Computation*. Mathematics and Its Applications. Kluwer, Dordrecht.

Richard G. Swan (1962). Factorization of polynomials over finite fields. *Pacific Journal of Mathematics* **12**, 1099–1106.

Alfred Wassermann (1990). Konstruktion von Normalbasen. *Bayreuther Mathematische Schriften* **31**, 155–164.

Alfred Wassermann (1993). Zur Arithmetik in endlichen Körpern. *Bayreuther Mathematische Schriften* **44**, 147–251.

Neal Zierler (1970). On $x^n + x + 1$ over $GF(2)$. *Information and Control* **16**, 502–505.

Neal Zierler and John Brillhart (1968). On primitive trinomials (Mod 2). *Information and Control* **13**, 541–554.

Neal Zierler and John Brillhart (1969). On primitive trinomials (Mod 2), II. *Information and Control* **14**, 566–569.