Software Implementation of Arithmetic in $\mathbb{F}_{3^m}$

Omran Ahmadi$^1$, Darrel Hankerson$^2$, and Alfred Menezes$^3$

1 Dept. of Electrical and Computer Engineering, University of Toronto
oahmadid@comm.utoronto.ca
2 Dept. of Mathematics and Statistics, Auburn University
hankedr@auburn.edu
3 Dept. of Combinatorics & Optimization, University of Waterloo
ajmeneze@uwaterloo.ca

Abstract. Fast arithmetic for characteristic three finite fields $\mathbb{F}_{3^m}$ is desirable in pairing-based cryptography because there is a suitable family of elliptic curves over $\mathbb{F}_{3^m}$ having embedding degree 6. In this paper we present some structure results for Gaussian normal bases of $\mathbb{F}_{3^m}$, and use the results to devise faster multiplication algorithms. We carefully compare multiplication in $\mathbb{F}_{3^m}$ using polynomial bases and Gaussian normal bases. Finally, we compare the speed of encryption and decryption for the Boneh-Franklin and Sakai-Kasahara identity-based encryption schemes at the 128-bit security level, in the case where supersingular elliptic curves with embedding degrees 2, 4 and 6 are employed.

1 Introduction

Pairing-based cryptographic protocols are realized using algebraic curves of low-embedding degree. Several families of low-embedding degree elliptic curves have been considered, including supersingular curves with embedding degrees 2, 4, and 6, and ordinary curves with embedding degrees 2 [29], 6 [23], and 12 [5]. The family of supersingular elliptic curves with embedding degree 6 is defined over characteristic three finite fields $\mathbb{F}_{3^m}$. Consequently, the software and hardware implementation of arithmetic in these fields has been intensively studied in recent years [16,25,13,20,14].

The elements of $\mathbb{F}_{3^m}$ can be represented using a polynomial basis or a normal basis. We present some new structure results for Gaussian normal bases of $\mathbb{F}_{3^m}$, and use these results to devise faster multiplication algorithms. Our implementation on a Pentium 4 machine shows that our fastest algorithm for normal basis multiplication in $\mathbb{F}_{3^{239}}$ is about 50% faster than standard Ning-Yin multiplication [24], and about 4.4 times faster than the Ning-Yin implementation reported by Granger, Page and Stam [14]. Our experiments also suggest that the comb method for polynomial basis multiplication [22] (perhaps combined with shallow-depth Karatsuba-like techniques) is faster than Karatsuba multiplication. In particular, our implementation for polynomial basis multiplication in $\mathbb{F}_{3^{239}}$ is about 4.6 times faster than that reported in [14]. We conclude, as in [14], that polynomial bases are preferred over normal bases for the software implementation of characteristic three field arithmetic.
A recent IETF draft standard \[8\] for identity-based encryption (IBE) mandates use of supersingular elliptic curves over prime fields — these curves have embedding degree 2. We compare the speed of encryption and decryption for the Boneh-Franklin \[7\] and Sakai-Kasahara \[27\] identity-based encryption schemes, when the underlying elliptic curves are supersingular and defined over a prime field (embedding degree 2), a characteristic two finite field (embedding degree 4), and a characteristic three finite field (embedding degree 6). We focus our attention on 1536-bit prime fields $\mathbb{F}_p$, the characteristic two field $\mathbb{F}_{2^{1223}}$, and the characteristic three field $\mathbb{F}_{3^{509}}$. Each of these choices achieves the 128-bit security level in the sense that the best attacks known on the discrete logarithm problem in the extension fields $\mathbb{F}_p^2$, $\mathbb{F}_{2^{4 \cdot 1223}}$, and $\mathbb{F}_{3^{6 \cdot 509}}$ have running time approximately $2^{128}$ \[21\].

We acknowledge that the Barreto-Naehrig (BN) ordinary elliptic curves \[5\] over 256-bit fields $\mathbb{F}_p$ with embedding degree 12 are ideally suited for pairing applications at the 128-security level, and can be expected to yield faster implementations than supersingular elliptic curves especially when the Ate pairing algorithm \[18\] is employed. However, some people are reluctant to use the BN curves because recent work by Schirokauer \[28\] has raised the possibility that the special number field sieve may be effective for computing discrete logarithms in $\mathbb{F}_{p^{12}}$. Furthermore, in some cases expensive hashing or the absence of an efficiently-computable isomorphism $\psi$ (cf. \[12\]) may be a concern. Thus it is worthwhile to consider the relative merits of supersingular elliptic curves in pairing-based cryptography.

The remainder of the paper is organized as follows. Methods for performing arithmetic in $\mathbb{F}_{3^m}$ using a polynomial basis representation are reviewed in \S 2. Our structure results for Gaussian normal bases are developed in \S 3. In \S 4 we present our implementation results for $\mathbb{F}_{3^m}$. Estimates for Boneh-Franklin and Sakai-Kasahara IBE are given in \S 5. Summary conclusions appear in \S 6.

## 2 Polynomial Bases for $\mathbb{F}_{3^m}$

Elements $a \in \mathbb{F}_{3^m}$ can be regarded as polynomials $a = a_{m-1}x^{m-1} + \cdots + a_0$ where $a_i \in \mathbb{F}_3$ and arithmetic is performed modulo an irreducible polynomial $f$ of degree $m$. We associate $a$ with the vector of coefficients $(a_{m-1}, \ldots, a_0)$.

Harrison, Page, and Smart \[16\] considered two coefficient representations suitable for implementation. Their “Type II” representation is closer to common techniques used for binary fields. Each coefficient $a_i \in \mathbb{F}_3$ is represented uniquely in $\{0, 1, -1\}$ using a pair $(a_i^0, a_i^1)$ of bits, where $a_i = a_i^0 - a_i^1$ and not both bits are 1. Elements $a$ are represented by vectors $a^j = (a^j_{m-1}, \ldots, a^j_0)$, $j \in \{0, 1\}$. Addition $c = a + b$ is

$$t \leftarrow (a^0 \lor b^1) \oplus (a^1 \lor b^0), \quad c^0 \leftarrow (a^1 \lor b^1) \oplus t, \quad c^1 \leftarrow (a^0 \lor b^0) \oplus t.$$  

The seven operations involve only bitwise “or” ($\lor$) and “exclusive-or” ($\oplus$), and it is easy to order the instructions to cooperate with processor pipelining. Negation is $-a = (a^1, a^0)$. 

...