A Note on Square Roots in Binary Fields

Roberto Maria Avanzi

Faculty of Mathematics and Horst Görtz Institute for IT-Security
Ruhr University Bochum, Germany
Roberto.Avanzi@ruhr-uni-bochum.de

Abstract. In this note we present a family of irreducible polynomials that can be used to speed up square root extraction in fields of characteristic two. This generalizes one of the families of trinomials presented by Fong et al. and the results are not limited to trinomials. In fact, we show for the first time pentanomials (polynomials with five nonzero terms) and eptanomials (polynomials with seven nonzero terms) allowing fast square root computation. We call such polynomials *square root friendly*. The obvious application is to point halving methods for elliptic and divisor halving methods for hyperelliptic curves.
We also note the existence of square root friendly trinomials of a given degree when we already know that an irreducible trinomial of the same degree exists, and formulate a conjecture on square root friendly pentanomials when an irreducible trinomial does not exist.

1 Introduction

This paper presents results on efficient square root computation in binary fields. In the seminal work [11] it is shown how to compute square roots very efficiently in polynomial basis representation when the odd degree field extension of $\mathbb{F}_2$ is defined by an irreducible trinomial. In this paper we show sufficient conditions for an irreducible polynomial of odd degree d to allow fast square root computation – we call such a polynomial *square root friendly*. In particular, our results are applied to pentanomials, but in at least one case, that of $\mathbb{F}_{2^{233}}$, that can be defined by trinomials, we show how one can perform square root computations even faster than in [11].

Since the motivation comes from elliptic curve cryptography, in particular from point halving based methods for scalar multiplication, we begin in Section 2 by recalling point and divisor halving and how square root computation comes into play. Then, in Section 3 our sufficient conditions are introduced. Square root friendly polynomials for several useful (and used in practice) binary fields are given in Section 4, together with a result about the existence of square root friendly trinomials, and a conjecture about square root friendly pentanomials. Some open questions round off the presentation.

2 Halving and Square Roots

2.1 Point and Divisor Halving

Let E be a generic elliptic curve defined over $\mathbb{F}_{2^d}$ by a *Weierstrass equation*

$$E \ : \ y^2 + xy = x^3 + ax^2 + b$$

with $a, b \in \mathbb{F}_{2^d}$ and having a subgroup $G \leq E(\mathbb{F}_{2^d})$ of large prime order. Formulæ to compute the double of any point P on the curve are well known, and since the doubling is the most common operation in a scalar multiplication performed by double-and-add methods, an important direction of research consists in optimizing these formulæ (see for example Chapters 9 and 13 of [4] for surveys on scalar multiplication and elliptic curve operations respectively).

Point halving [12, 17], on the other hand, consists in computing a point R whose double is P, i.e. such that $2R = P$. To a point P with affine coordinates (x, y) we associate the quantity $\lambda_P = x + \frac{y}{x}$. Let $P = (x, y)$ and $R = (u, v)$ be points of $E(\mathbb{F}_{2^d}) \setminus \{0\}$ with $2R = P$. The affine coordinates of P and R are related as follows:

$$\lambda_R = u + \frac{v}{u} \tag{1}$$

$$x = \lambda_R^2 + \lambda_R + a \tag{2}$$

$$y = u^2 + x(\lambda_R + 1) \tag{3}$$

In order find R we have to solve (2) for λ, (3) for u, and finally (1) for v. After some simple manipulations, we see that we have to perform the following operations:

$$\text{(i)} \quad \text{Solve} \ \ \lambda_R^2 + \lambda_R = a + x \ \ \text{for} \ \ \lambda_R \tag{4}$$

$$\text{(ii)} \quad \text{Put} \ \ t = y + x(\lambda_R + 1)$$

$$\text{(iii)} \quad \text{Find} \ \ u \ \ \text{with} \ \ u^2 = t \tag{5}$$

$$\text{(iv)} \quad \text{Put} \ \ v = t + u\lambda_R \ .$$

Point halving, being the inverse operation of the doubling, is an automorphism of G. Therefore, given a point $P \in G$, there is a unique $R \in G$ such that $2R = P$. In other words, the equations (4) and (5) can always be solved in $\mathbb{F}_{2^d}$. But, they do not determine a *unique* point R with $2R = P$. In fact, solving them will always yield two distinct points R_1 and R_2 such that $R_1 - R_2$ is the unique point of order 2 of the curve. It is possible,

by performing an additional check, to determine the point $R \in G$. Knudsen [12] and Schroeppel [16, 17] show how to perform all these steps and checks in an efficient way. According to the very thorough analysis in [11], halving is about two times faster than doubling. We refer the interested reader to [12, 16, 17, 11] for details, including methods to use the halving in place of the doubling in scalar multiplication algorithms.

Birkner [8] has found a divisor halving formula for genus two curves based on the doubling formulae by Lange and Stevens [14]. Birkner and Thériault [9] have also tackled halving for genus three divisors. The performance of these halving formulæ also depends on the performance of square root extraction, but to a smaller degree than on elliptic curves.

Further uses of point halving to speed up scalar multiplication on the special class of elliptic Koblitz Curves [13] are found in [3, 5] and [6].

2.2 Square Root Computation

In what follows will be only concerned with square root extraction for binary fields represented via a polynomial basis: Let $p(X)$ be an irreducible polynomial of *odd* degree d. The field $\mathbb{F}_{2^d}$ be constructed as the quotient ring $\mathbb{F}_2[X]/(p(X))$, and we identify X with its own image in this ring.

The reason behind this is that in software applications it is customary to use a polynomial basis representation for the field extension $\mathbb{F}_{2^d}/\mathbb{F}_2$ instead of a normal basis [2], because in the latter representation the cost of a software multiplication is much higher than with a polynomial basis.

But, whereas with normal basis a square root computation is just a shift of the bits internal representation of the field element by one position, matters are more complicated with polynomial bases.

In fact, even the cost of a squaring is not completely negligible with polynomial bases. If

$$\alpha = \sum_{i=0}^{d-1} a_i X^i$$

then

$$\alpha^2 = \sum_{i=0}^{d-1} a_i X^{2i}$$

which, as a polynomial in X, has degree no longer necessarily bounded by d, and modular reduction modulo $p(X)$ is necessary. Its cost is very low, but not completely negligible.

Things are a bit more complicated for square root computation. In fact, squaring just consists in "spacing" the bits of the original element with zeros, but the bits of a generic element cannot be just "squeezed".

The classic method for computing $\sqrt{\alpha}$, is based on Fermat's little theorem $\alpha^{2^d} = \alpha$, hence $\sqrt{\alpha} = \alpha^{2^{d-1}}$. This requires $d-1$ squarings. In general, the cost of this operation is worse than several field multiplications.

A more efficient method stems from the observation that $\sqrt{\alpha}$ can be expressed in terms of $\zeta := \sqrt{X}$. If

$$\alpha = \sum_{i=0}^{d-1} a_i X^i$$

we separate the even exponents from the odd exponents

$$\alpha = \sum_{i=0}^{\frac{d-1}{2}} a_{2i} X^{2i} + \sum_{i=0}^{\frac{d-3}{2}} a_{2i+1} X^{2i+1} = \sum_{i=0}^{\frac{d-1}{2}} a_{2i} X^{2i} + X \cdot \sum_{i=0}^{\frac{d-3}{2}} a_{2i+1} X^{2i}$$

and, since square root extraction in fields of even characteristic is a linear operation:

$$\sqrt{\alpha} = \sum_{i=0}^{\frac{d-1}{2}} a_{2i} X^i + \zeta \cdot \sum_{i=0}^{\frac{d-3}{2}} a_{2i+1} X^i \ . \tag{6}$$

Therefore, once ζ has been computed on a per-field basis, the computation of a generic square root is reduced to "bits extraction and packing", a "rectangular" multiplication of a degree $\leq d-1$ polynomial ζ with a polynomial $\sum_{i=0}^{(d-3)/2} a_{2i+1} X^i$ of degree $\leq (d-1)/2$, and a modular reduction. Intuitively, the cost should approach a half of the cost of a field multiplication, and this is confirmed by the analysis in [11, 12].

3 New Defining Polynomials

As we have just seen, efficient square root computation depends on the efficiency of the multiplication of a generic degree $\leq (d-1)/2$ polynomial by $\zeta = \sqrt{X}$. If ζ is a very sparse element, for example of weight two or four (i.e. it has just two or four nonzero terms), then this product can be computed by a few shift and xor operations. In [11] two types of trinomials have been shown that allow this. The kind that interests us is

$$p(X) = X^d + X^m + 1$$

with m odd. Then $X = X^{d+1} + X^{m+1}$ with $d+1$ and $m+1$ even, and

$$\zeta = X^{(d+1)/2} + X^{(m+1)/2} \ ,$$

and $p(X)$ is square root friendly. In fact, this idea is much more general.

Assume we have an irreducible polynomial $p(X)$ defining $\mathbb{F}_{2^d}$ over $\mathbb{F}_2$ of the form

$$p(X) = X \cdot \mathcal{U}(X)^2 + 1 \tag{7}$$

where $\mathcal{U}$ is a polynomial of degree $(d-1)/2$ and even weight. Then, ζ has a very simple form in $\mathbb{F}_{2^d}$: from

$$X^2 \cdot \mathcal{U}(X)^2 + X = 0$$

we obtain

$$\zeta = X \cdot \mathcal{U}(X) \ .$$

Note at this point that the *polynomial* product

$$\zeta \cdot \sum_{i=0}^{\frac{d-3}{2}} a_{2i+1} X^i$$

has degree bounded by $\left(1+\frac{d-1}{2}\right)+\frac{d-3}{2} = d$, therefore *polynomial reduction is very easy*, because at most one single bit (corresponding to a term X^d) has to be reduced. Hence, irreducible polynomials $p(X)$ of form (7) are square root friendly.

Examples of such polynomials abound. For example $X^{163} + X^{65} + X^{35} + X^{33} + 1$ is irreducible, and under this representation ζ has weight 4. On the other hand, the standard NIST polynomial [15] $X^{163} + X^7 + X^6 + X^3 + 1$ defines a ζ of weight 79. Changing polynomial is in fact easy without introducing incompatibilities in the practical use: we just change the base used for representation of the field elements before and after the whole scalar multiplication. The cost is comparable to a polynomial basis multiplication, and the conversion routines require each a matrix that occupies $O(d^2)$ bits of storage (see for instance [10], where the particular base change is to and from a normal basis representation, but the results are the same). Therefore this overhead is essentially negligible with respect to the full cost of the operation that is in the order of magnitude of thousands of field multiplications. The bulk of the computation is then performed in the "easy" representation, the inputs and outputs are given in the "standard" representation.

The cost of a square root extraction implemented by using the sparse version of ζ offered by the above polynomials can be roughly estimated

using, for example, already published results. For example in [11], Example 3.12, the NIST-recommended trinomial $p(X) = X^{233} + X^{74} + 1$ for the finite field $\mathbb{F}_{2^{233}}$ is used. Even though the term X^{74} does not have an even exponent, ζ has a sparse representation

$$\zeta = (X^{32} + X^{117} + X^{191})(X^{37} + 1) \ .$$

By means of this representation, finding a root via equation (6) requires roughly 1/8 of the time of a field multiplication. As reported in the next section we can choose also $p(X) = X^{233} + X^{159} + 1$ for which

$$\zeta = X^{117} + X^{80} \ .$$

In this case it is clear that much fewer shift operations and XOR operations are required to multiply by ζ. Furthermore, as already remarked, there is no need to perform a reduction modulo $p(X)$ as complex as with the standard polynomial. First implementation results show the cost of a square root to be about 8% of that of a multiplication.

Similar formulæ for cube root computations are found in [1] – their results are easily partially generalised to any odd characteristic.

4 Existence and other Properties

There is plenty of square root friendly polynomials. For example, for extension degree $d = 163$, a simple computer program immediately yields several examples. In Table 1 square root friendly polynomials of several different degrees can be found. The degrees have been taken from the NIST list of recommended binary curves and from the extension degrees used in [7]. When no trinomial is available, a pentanomial is used. We always report the polynomial with least degree sediment (the sediment of an univariate polynomial is the polynomial itself with the leading term removed). In particular, observe that also efficient trinomials are available. All these extension degrees are interesting because they are either used in standards for elliptic curve cryptography or they represent good choices for extension degrees for defining hyperelliptic curve for cryptographic applications. Only in a handful of cases is the square root friendly polynomial with least degree sediment the same as the standard one, i.e. the irreducible polynomial with least degree sediment but without the restriction on being square root friendly.

Theorem. *Let d be an odd positive integer. If an irreducible trinomial $p(X)$ over $\mathbb{F}_2$ of degree d exists, then $p(X)$ can be chosen of the form* (7)*, i.e. where all the non-vanishing exponents are odd.*

Degree	Irreducible polynomial	$\zeta = \sqrt{X}$	Standard?
47	$X^{47} + X^5 + 1$	$X^{24} + X^3$	Yes
53	$X^{53} + X^{19} + X^{17} + X^{15} + 1$	$X^{27} + X^{10} + X^9 + X^8$	No
59	$X^{59} + X^{21} + X^{17} + X^{15} + 1$	$X^{30} + X^{11} + X^9 + X^8$	No
67	$X^{67} + X^{25} + X^{17} + X^5 + 1$	$X^{34} + X^{13} + X^9 + X^3$	No
71	$X^{71} + X^9 + 1$	$X^{36} + X^5$	No
73	$X^{73} + X^{25} + 1$	$X^{37} + X^{13}$	Yes
79	$X^{79} + X^9 + 1$	$X^{40} + X^5$	Yes
83	$X^{83} + X^{29} + X^{25} + X^3 + 1$	$X^{42} + X^{15} + X^{13} + X^2$	No
89	$X^{89} + X^{51} + 1$	$X^{45} + X^{26}$	No
97	$X^{97} + X^{33} + 1$	$X^{49} + X^{17}$	No
101	$X^{101} + X^{35} + X^{31} + X^3 + 1$	$X^{51} + X^{18} + X^{16} + X^2$	No
107	$X^{107} + X^{37} + X^{33} + X^{23} + 1$	$X^{54} + X^{19} + X^{17} + X^{12}$	No
109	$X^{109} + X^{43} + X^{41} + X^{23} + 1$	$X^{55} + X^{22} + X^{21} + X^{12}$	No
127	$X^{127} + X + 1$	$X^{64} + X$	Yes
131	$X^{131} + X^{45} + X^{41} + X^9 + 1$	$X^{66} + X^{23} + X^{21} + X^5$	No
137	$X^{137} + X^{21} + 1$	$X^{69} + X^{11}$	Yes
139	$X^{139} + X^{53} + X^{33} + X^{25} + 1$	$X^{70} + X^{27} + X^{17} + X^{13}$	No
149	$X^{149} + X^{51} + X^{47} + X^9 + 1$	$X^{75} + X^{26} + X^{24} + X^5$	No
157	$X^{157} + X^{55} + X^{47} + X^{11} + 1$	$X^{79} + X^{28} + X^{24} + X^6$	No
163	$X^{163} + X^{57} + X^{49} + X^{29} + 1$	$X^{82} + X^{29} + X^{25} + X^{15}$	No
179	$X^{179} + X^{61} + X^{57} + X^{41} + 1$	$X^{90} + X^{31} + X^{29} + X^{21}$	No
199	$X^{199} + X^{67} + 1$	$X^{100} + X^{34}$	No
211	$X^{211} + X^{73} + X^{69} + X^{35} + 1$	$X^{106} + X^{37} + X^{35} + X^{18}$	No
233	$X^{233} + X^{159} + 1$	$X^{117} + X^{80}$	No
239	$X^{239} + X^{81} + 1$	$X^{120} + X^{41}$	No
251	$X^{251} + X^{89} + X^{81} + X^3 + 1$	$X^{126} + X^{45} + X^{41} + X^2$	No
269	$X^{269} + X^{91} + X^{87} + X^{61} + 1$	$X^{135} + X^{46} + X^{44} + X^{31}$	No
283	$X^{283} + X^{97} + X^{89} + X^{87} + 1$	$X^{142} + X^{49} + X^{45} + X^{44}$	No
409	$X^{409} + X^{87} + 1$	$X^{205} + X44$	Yes
571	$X^{571} + X^{193} + X^{185} + X^5 + 1$	$X^{286} + X^{97} + X^{93} + X^3$	No

Table 1. Square root friendly irreducible polynomials.

Proof. Let

$$X^d + X^m + 1$$

be an irreducible trinomial with $d > m > 0$ and m even. Then it is easy to prove that the polynomial

$$X^d + X^{d-m} + 1$$

is also irreducible – but $d - m$ is odd. In fact, let $q(X)$ be a monic polynomial over $\mathbb{F}_2$ with $q(X) = 1$, i.e. non-vanishing constant term. Define

$$\hat{q}(X) = X^{\deg q} q(X^{-1})$$

to be the *inversion* of $q(X)$. It ie easy to see that $\hat{q}(X)$ is a monic polynomial with non-vanishing constant term. Then, a factorization $q(X) = g(X)h(X)$ implies $\hat{q}(X) = \hat{g}(X)\hat{h}(X)$. Applying this result to $q(X) = X^d + X^{d-m} + 1$ proves that it must be irreducible, otherwise $p(X) = \hat{q}(X)$ would be reducible, too. □

Existence results for pentanomial-defined fields are still an open question. However, on the basis of the above table and further experimental results, we formulate the following conjecture:

Conjecture. *Let d be an odd prime integer such that there exist no irreducible trinomials of degree d over $\mathbb{F}_2$, but irreducible pentanomials of degree d exist. Then irreducible pentanomials of the form*

$$X^d + X^c + X^b + X^a + 1$$

where $d > c > b > a > 0$, and a, b, c are odd, also exist. Furthermore, let $p(X)$ be such a polynomial with c smallest. Then $c \approx d/3$.

Further open questions are: how to find polynomials like the above that are also efficient for almost-inverse computations; to balance the possibly increased modular reduction cost with the savings obtained in other parts of the computations. These will be the subject of future work.

Acknowledgement. *The author is grateful to Darrel Hankerson and Alfred Menezes for fruitful conversations on the matter in the month of March, 2005, as well as to Nicolas Thériault and Peter Birkner for further discussions on the matter during a stay at the Fields Institute, Toronto, in September, 2006.*

References

1. O. Ahmadi, D. Hankerson, and A. Menezes. *Formulas for cube roots in $\mathbb{F}_{3^m}$.* Discrete Applied math. **155** (3), 260–270, 2007.
2. D. W. Ash, I. F. Blake and S. Vanstone. *Low complexity normal bases.* Discrete Applied Math. **25** 191–210, 1989.
3. R. M. Avanzi, M. Ciet, and F. Sica. *Faster Scalar Multiplication on Koblitz Curves combining Point Halving with the Frobenius Endomorphism.* Proceedings of PKC 2004, LNCS **2947**, 28–40. Springer–Verlag, 2004.
4. R. Avanzi, H. Cohen, C. Doche, G. Frey, T. Lange, K. Nguyen, and F. Vercauteren. *The Handbook of Elliptic and Hyperelliptic Curve Cryptography.* CRC Press, 2005.
5. R. M. Avanzi, C. Heuberger, and H. Prodinger. *Scalar Multiplication on Koblitz Curves Using the Frobenius Endomorphism and its Combination with Point Halving: Extensions and Mathematical Analysis.* Algorithmica **46** (2006), 249–270
6. R. M. Avanzi, C. Heuberger, and H. Prodinger. *On Redundant τ-adic Expansions and Non-Adjacent Digit Sets.* To appear in proceedings of SAC 2006.
7. R. M. Avanzi, N. Thériault, and Z. Wang. *Rethinking Low Genus Hyperelliptic Jacobian Arithmetic over Binary Fields: Interplay of Field Arithmetic and Explicit Formulæ.* CACR Technical Report 2006-07.
8. P. Birkner: *Efficient Divisor Class Halving on Genus Two Curves.* To appear in: Proceedings of Selected Areas in Cryptography – SAC 2006. Springer Verlag LNCS.
9. P. Birkner, and N. Thériault. *Efficient Divisor Class Doubling and Halving on Genus Three Curves.* In preparation.
10. J.-S. Coron, D. M'Raïhi, and C. Tymen. *Fast generation of pairs $(k, [k]P)$ for Koblitz elliptic curves.* In *Proceedings of SAC 2001*, Lecture Notes in Computer Science 2259, pp. 151–164. Springer, 2001.
11. K. Fong, D, Hankerson, J, López, A, Menezes:. Field Inversion and Point Halving Revisited. IEEE Trans. Computers 53(8), 1047–1059, 2004.
12. E. W. Knudsen. *Elliptic Scalar Multiplication Using Point Halving.* Proocedings of ASIACRYPT 1999, LNCS 1716, 135–149. Springer, 1999.
13. N. Koblitz. *CM-curves with good cryptographic properties.* In: *Proceedings of CRYPTO 1991*, LNCS 576, pp. 279–287. Springer, 1991.
14. T. Lange and M. Stevens. Efficient doubling for genus two curves over binary fields. In: *Selected Areas in Cryptography – SAC 2004.* LNCS 3357, 170–181, Springer-Verlag, 2005.
15. National Institute of Standards and Technology. Recommended Elliptic Curves for Federal Government Use. NIST Special Publication, July 1999.
    Available from: `http://csrc.nist.gov/csrc/fedstandards.html`
16. R. Schroeppel. *Point halving wins big.* Talks at: (i) Midwest Arithmetical Geometry in Cryptography Workshop, November 17–19, 2000, University of Illinois at Urbana-Champaign; and (ii) ECC 2001 Workshop, October 29–31, 2001, University of Waterloo, Ontario, Canada.
17. R. Schroeppel. *Elliptic curve point ambiguity resolution apparatus and method.* International Application Number PCT/US00/31014, filed 9 November 2000.