

Function Field Sieve in Characteristic Three

R. Granger¹, A.J. Holt², D. Page¹, N.P. Smart¹, and F. Vercauteren¹

¹ Dept. Computer Science,
University of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB,
United Kingdom.

{granger,page,nigel,frederik}@cs.bris.ac.uk

² Dept. Computer Science,
University of Bath,
Bath, BA2 7AY,
United Kingdom.
ajh@cs.bath.ac.uk

Abstract. In this paper we investigate the efficiency of the function field sieve to compute discrete logarithms in the finite fields \mathbb{F}_{3^n} . Motivated by attacks on identity based encryption systems using supersingular elliptic curves, we pay special attention to the case where n is composite. This allows us to represent the function field over different base fields. Practical experiments appear to show that a function field over \mathbb{F}_3 gives the best results.

1 Introduction

Research into the discrete logarithm problem (DLP) in finite fields has typically only focused on characteristic two and large prime fields, because these offer the greatest efficiency for implementations. However, since the proposal of the first fully functional identity based encryption scheme (IBE) by Boneh and Franklin [4], characteristic three fields have gained a significant cryptographic interest. These fields allow optimal security parameters with reduced bandwidth in the context of supersingular elliptic curves and pairings [9], used by concrete IBE implementations.

First suggested by Shamir [22] in 1984, the concept of identity based cryptography has been an attractive target for researchers because it has the potential to massively reduce the complexity of current PKI systems. By using the notion of identity as a users public key, the amount of infrastructure required is greatly reduced since a message sender implicitly knows the public key of the recipient. An often used example is that of an email address. Within this context, an identity for Alice might be the string

alice@hotmail.com.

If Bob wants to send Alice secure email, he implicitly knows her email address and hence her identity and public key. Therefore he can encrypt the email to her without the same level of involvement from, for example, certificate and trust authorities.

Since the paper of Boneh and Franklin [4] a large body of work on efficient implementation of the underlying algorithms [5, 10] and arithmetic in both software [13] and hardware [6, 19], has made such applications viable. Almost all these implementations have made use of elliptic curves defined over finite fields of characteristic three. The security of these IBE implementations is therefore dependent on the difficulty of solving the discrete logarithm problem both on an elliptic curve over a field of characteristic three and in a finite field of characteristic three.

Algorithms for discrete logarithm problems on elliptic curves do not depend, essentially, on the underlying finite field. Hence, their behaviour is well understood. The same cannot be said for discrete logarithm algorithms over finite fields. Such algorithms depend quite closely on the characteristic. Indeed different algorithms are chosen for different size characteristics.

The best known algorithm for solving the DLP in fields \mathbb{F}_{p^n} of small characteristic is currently the Function Field Sieve (FFS). First developed by Adleman [1] and Adleman and Huang [2] in 1994, the FFS may be regarded as an extension of Coppersmith's earlier discrete logarithm algorithm for characteristic two fields [7, 8]. Initially the algorithm applied to small fields subject to the condition $p^6 \leq n$ as $p^n \rightarrow \infty$, where p^n is the size of the finite field. Schirokauer [21] then extended the method to relax the condition to $p \leq n^{o\sqrt{n}}$. Asymptotically it has complexity

$$L_{p^n}[1/3, (32/9)^{\frac{1}{3}}] = \exp\left(\left((32/9)^{\frac{1}{3}} + o(1)\right) \log(p^n)^{\frac{1}{3}} \log(\log(p^n))^{\frac{2}{3}}\right).$$

More recently, Joux and Lercier [14] provided a slightly more practical algorithm, which chooses the initial function field and resulting polynomials more efficiently. It is this latter method that we adopt for our implementation making minor changes to facilitate the use of superelliptic curves which provide some efficiency benefits.

In the past discrete logarithm computations in finite fields of small characteristic have been carried out only in characteristic two, see for example [11, 24, 14]. In each of these works the extension degree was prime, since the underlying protocols being attacked were assumed to come from traditional discrete logarithm based systems in which one almost always selects prime extension degrees.

The fields of characteristic three arising in identity based encryption systems are all of composite extension degree. The so called MOV embedding is usually chosen to be six, see [18, 9]. We then have the option to apply the FFS method relative to any subfield. A theoretical analysis indicates that in this case, some representations may be more amenable to attack than others [12], and we investigate these possibilities also.

The remainder of this paper is organized as follows: in Section 2 we recall the mathematical background of the function field sieve and in Section 3 we provide

a detailed description of our implementation. In Section 4 we report on several practical experiments which investigate the efficiency of the function field sieve in characteristic three and in Section 5 we present some conclusions.

2 The Function Field Sieve

The finite fields of interest to pairing based cryptography in characteristic three are given by $\mathbb{K} = \mathbb{F}_{3^n}$ where $n = 6 \cdot m$. In what follows we set $q = 3^m$. In practice, we limit ourselves to finite fields which could arise as the MOV embedding of a supersingular elliptic curve over the field \mathbb{F}_q whose group order is divisible by a prime l of size comparable to that of q . Such elliptic curves have group orders given by

$$q \pm \sqrt{3q} + 1.$$

We wish to investigate not only the practicality of the function field sieve for the field extension $\mathbb{F}_{3^n}/\mathbb{F}_3$, but also the effect of taking different base fields, i.e. looking at the extension $\mathbb{F}_{3^n}/\mathbb{F}_{3^e}$ where $e = 1, 2, 3$ or 6 . To this end we let $k = \mathbb{F}_{3^e}$, $N = n/e$ and $p = 3^e$.

We assume we are given $\alpha, \beta \in \mathbb{K}$, both of order l , such that

$$\beta = \alpha^x$$

for some unknown $x \in \{1, \dots, l-1\}$. The discrete logarithm problem then is to compute x given α and β .

We will use a function field $F = k(X)[Y]/(H)$ defined by the polynomial $H(X, Y) \in k[X, Y]$ over the rational function field $k(X)$. Note, we shall abuse notation slightly and refer to the polynomial $H(X, Y)$ as the curve H , by which we mean the curve defined by $H(X, Y) = 0$. For practical reasons [17], one usually selects a C_{ab} -curve for H . However, in our examples we used a superelliptic curve of the form

$$H(X, Y) = Y^d + R(X)$$

where $R(X)$ is a polynomial in $k[X]$ of degree b . This enables more efficient calculation of the functions on the algebraic side, at the expense of a little less generality of our implementation. The class number of the function field F defined by H , or equivalently the number of points on the Jacobian of the curve J_H defined by H over k , we shall denote by $h = \#J_H(k)$.

We assume that the field \mathbb{K} is defined by a polynomial basis with respect to the polynomial $f \in k[X]$ of degree N , i.e.

$$\mathbb{K} \cong k[X]/(f).$$

To define the function field sieve algorithm we need to specify two polynomials $u_1, u_2 \in k[X]$ such that the norm of the function $u_1 + u_2 Y$, given by the resultant

$$u_2^d H(X, -u_1/u_2) = (-u_1)^d + u_2^d R$$

is divisible by f . In such a situation we have a surjective homomorphism

$$\phi : \begin{cases} k[X, Y]/(H) & \longrightarrow \mathbb{K} \cong k[X]/(f) \\ Y & \mapsto -u_1/u_2. \end{cases}$$

We select a rational factorbase \mathcal{R} of small degree irreducible polynomials in $k[X]$ and an algebraic factorbase \mathcal{A} of small prime divisors in the divisor group $\text{Div}(F)$. Hence, \mathcal{R} and \mathcal{A} are therefore defined by

$$\begin{aligned} \mathcal{R} &= \{ \mathfrak{p} : \deg \mathfrak{p} \leq B, \mathfrak{p} \text{ irreducible} \}, \\ \mathcal{A} &= \{ \langle \mathfrak{p}, Y - \mathfrak{r} \rangle : \deg \mathfrak{p} \leq B, \mathfrak{p} \text{ irreducible}, \mathfrak{r} \equiv -u_1/u_2 \pmod{\mathfrak{p}} \}, \end{aligned}$$

for some smoothness bound B .

The goal of the function field sieve is to find relatively prime pairs of polynomials (r, s) , with $\deg r, \deg s \leq l$, such that the polynomial $(su_2 - ru_1)$ and the divisor $\langle s + rY \rangle$ simultaneously factor over the respective factorbases, i.e.

$$\begin{aligned} su_2 - ru_1 &= \prod_{\mathfrak{p}_i \in \mathcal{R}} \mathfrak{p}_i^{a_i} \\ \langle s + rY \rangle &= \sum_{\langle \mathfrak{p}_j, Y - \mathfrak{r}_j \rangle \in \mathcal{A}} b_j \langle \mathfrak{p}_j, Y - \mathfrak{r}_j \rangle. \end{aligned}$$

Determining the factorization of $\langle s + rY \rangle$ over \mathcal{A} is easily done by examining the factorization of the norm

$$N(\langle s + rY \rangle) = (-s)^d + r^d R.$$

Since $h \langle \mathfrak{p}_j, Y - \mathfrak{r}_j \rangle$ is a principal divisor, for each $\langle \mathfrak{p}_j, Y - \mathfrak{r}_j \rangle \in \mathcal{A}$ there exists a function $\lambda_j \in F^*$ with $h \langle \mathfrak{p}_j, Y - \mathfrak{r}_j \rangle = \langle \lambda_j \rangle$ and such a function is unique up to multiplication by an element in k^* . We then have that our algebraic relation is given by

$$(s + rY)^h = \mu \prod_{\lambda_j} \lambda_j^{b_j},$$

with μ an element in k^* . We then apply the homomorphism ϕ above to obtain

$$\left(s - r \frac{u_1}{u_2} \right)^h \equiv \prod_{\lambda_j} \phi(\lambda_j)^{b_j},$$

where \equiv denotes equality modulo a possible factor in k^* . Assuming h is coprime to $(p^N - 1)/(p - 1)$, we can take h -th roots of both sides of this equation and if we write $\kappa_j = \phi(\lambda_j)^{1/h}$ we obtain

$$\left(s - r \frac{u_1}{u_2} \right) \equiv \prod_{\lambda_j} \kappa_j^{b_j}.$$

Combining the relation on the rational side with the above equation we find

$$\frac{1}{u_2} \prod_{\mathfrak{p}_i \in \mathcal{R}} \mathfrak{p}_i^{a_i} \equiv \prod_{\lambda_j} \kappa_j^{b_j}.$$

Hence, we obtain the relation between discrete logarithms given by

$$\sum_{\mathfrak{p}_i} a_i \log_g \mathfrak{p}_i - \log_g(u_2) = \sum_{\gamma_j} b_j \log_g \kappa_j, \quad (1)$$

where g is a multiplicative generator of the field \mathbb{K} . Note that we do not need at any point to compute the values of the κ_j , or for that matter the values of the λ_j , all that we require is that they exist. This is guaranteed by the condition that h should be coprime to $(p^N - 1)/(p - 1)$.

If sufficiently many independent relations of the form (1) have been obtained, we can solve for the discrete logarithms themselves using structured Gaussian elimination combined with the Lanczos method. Determining the discrete logarithm of β with respect to α is then performed using a standard recursive sieving strategy as explained in [21].

3 Choice of Parameters and Implementation Details

The various parameters of the function field sieve, namely the size d of the function field extension, the size B of the largest factorbase element and the size l of the polynomials r and s , are approximated by a heuristic analysis [14] of the function field sieve as

$$\begin{aligned} l &= B, \\ B &= \left\lceil \left(\frac{4}{9}\right)^{1/3} N^{1/3} \log_p(N)^{2/3} \right\rceil, \\ d &= \left\lceil \sqrt{\frac{N}{B+1}} \right\rceil. \end{aligned}$$

Since we have restricted ourselves to superelliptic curves we need to ignore values of d which are divisible by three. However, in the range of our experiments this is not an issue and extending our results to the case where $d \equiv 0 \pmod{3}$ can be accomplished using C_{ab} -curves [17], at the expense of more complicated formulae on the algebraic side.

3.1 Selection of f

Following Joux and Lercier [14] we first select u_1 and u_2 and then find a suitable value of f . We set $N' = N \pmod{d}$. However, since the degree $N = n/e$ may not be prime we split into three possible sub-cases:

- Case (i) : $\gcd(N', d) = 1$.

We choose a curve H such that $R(X)$ is of degree $b = N'$ and such that $\#J_H(k)$ is coprime to $(p^N - 1)/(p - 1)$. If this is not possible we go to Case (iii). We then select u_1 and u_2 of exact degrees $m - 1$ and m respectively where $m = (N - b)/d$. The degree of the polynomial $f(x) = u_2^d H(X, -u_1/u_2)$ is then given by $\max\{d \cdot (m - 1), d \cdot m + b\} = N$. Hence, we keep selecting u_1 and u_2 until f is irreducible.

- Case (ii) : $N' = 0$.
We select $R(X)$ of smallest possible degree b such that $\#J_H(k)$ is coprime to $(p^N - 1)/(p - 1)$. We then select u_1 and u_2 of exact degrees m and $m - 1$ where $m = N/d$. The degree of the polynomial $f(x) = u_2^d H(X, -u_1/u_2)$ is then given by $\max\{d \cdot m, d \cdot (m - 1) + b\} = N$, assuming $\deg R < d$. Hence, we keep selecting u_1 and u_2 until f is irreducible.
- Case (iii) : $\gcd(N', d) \neq 1, d$, or no suitable curve found above.
Here we select $R(X)$ of degree $b < d$, with $\gcd(b, d) = 1$, such that $\#J_H(k)$ is coprime to $(p^N - 1)/(p - 1)$ and for which one of t_1 and t_2 is minimal where

$$t_1 = \max\{N, d \cdot m, d \cdot (m - 1) + b\},$$

$$t_2 = \max\{N, d \cdot (m - 1), d \cdot m + b\}.$$

In the case of t_1 being minimal we then select u_1 and u_2 of degree m and $m - 1$ until $u_2^d H(X, -u_1/u_2)$ is divisible by an irreducible polynomial f of degree N . In the case of t_2 being minimal we select u_1 and u_2 to be of degree $m - 1$ and m , until we obtain an irreducible polynomial f of degree N which divides $u_2^d H(X, -u_1/u_2)$.

Note that Joux and Lercier [14] only considered Case (i) above, since they used arbitrary C_{ab} -curves (and not simply superelliptic ones), and because the extension degree N was always prime. Clearly, Case (iii) will lead to marginally less efficient relation collection, since the degree of the algebraic side will be slightly higher than if one was in Case (i) or (ii). However, if one is to deal with non-prime values of N and superelliptic curves one is led to such a case.

3.2 Lattice Sieving

The finding of (r, s) is performed using a lattice sieve. In the lattice sieve one selects a prime polynomial $\mathfrak{p} \in \mathcal{R}$ (resp. prime divisor $\langle \mathfrak{p}, Y - \mathfrak{r} \rangle \in \mathcal{A}$). Then one looks at the sub-lattice of the (r, s) plane such that $su_2 - ru_1$ (resp. $\langle s + rY \rangle$) is divisible by the chosen polynomial (resp. divisor). We found it more efficient to then sieve in the sub-lattice on a line-by-line basis, rather than sieve in a two-dimensional manner in the sub-lattice.

The use of lattice sieving has a number of advantages over sieving in the (r, s) plane. Firstly, it is better at yielding a large number of relations. Secondly, one can target factorbase elements for which one does not yet have a relation. This enables one to obtain a matrix involving all elements in the factorbase reasonably efficiently. Thirdly, the use of lattice sieving is crucial in the final stage where one wishes to target individual discrete logarithms using the recursive sieving strategy mentioned earlier.

3.3 Factorbase Size

A major consideration is the balancing of the effort required for the sieving and the matrix step. In theory one balances the two halves of the computation

and so derives the complexity estimates such as those above. The size of both factorbases is approximated by p^B/B , however one should treat this size as a continuous function of a real variable B , as opposed to the discrete B above. See [12] for further analysis of this point.

However, sieving can be performed in a highly scalable manner. After all, to move from using a single computer performing the sieving to around 100-200 computers is relatively easy given the resources of most organizations these days. However, our code for the matrix step required the use of a single machine and hence does not scale.

Hence, in practice we can devote less total time to the matrix step compared to the sieving step. Since the matrix step has complexity approximately $O(T^2)$ where $T = \#\mathcal{R} + \#\mathcal{A} \approx 2p^B/B$, we see that we have a physical constraint on the size of the factorbases we can accommodate. In our experiments we assumed that solving a matrix with over half a million rows and columns was infeasible given our resources and matrix code. This sometimes led us to choose non-optimal, from a theoretical perspective, values for the other parameters.

3.4 Linear Algebra Step

Several studies involving use of index calculus-type methods discuss the linear algebra step, since it is a major practical bottleneck in the procedure. This is because parallelising existing algorithms is rather difficult. Various authors [15, 20], have identified several effective techniques for the solution of sparse linear systems modulo a prime. These include iterative schemes such as the Lanczos, Conjugate Gradient and Wiedemann algorithms, with or without a pre-processing step involving structured Gaussian elimination. Recently attention has moved onto attempting to perform this step in parallel, see for example [25] for the case of Lanczos over the field \mathbb{F}_2 .

In our implementation we followed [15] and used a basic structured Gaussian elimination routine, such as that described in [3, 15, 20], so as to reduce the size of the linear system whilst maintaining a degree of sparsity. This submatrix is subsequently solved by the Lanczos algorithm [16], and a full result set is then recovered via back-substitution. Our implementation for this stage made use of Victor Shoup's NTL C++ library for multiprecision integers [23].

4 Experiments

As mentioned in the introduction, we are not only interested in how efficient the function field sieve method is on fields of relevance to IBE based systems. We are also interested in the effect of choosing a different base field $k = \mathbb{F}_{3^e}$ in the function field sieve. Usually for traditional discrete logarithm systems this would not be an issue since the extension degree is often prime. One should note that a precise theoretical analysis [12] of the function field sieve method shows that for a fixed field size the effect of the size of the base field is not as one would expect. This strange effect of the base field size 3^e on the overall performance

for a fixed field size 3^n can be explained due to the non-continuous behaviour of various parameters, in particular the function field extension degree d .

Note, the expected run time would be T^2 operations in $k[x]$. Each field operation requires at most $O(e^2)$ operations in \mathbb{F}_3 and the polynomials involved are of degree around

$$\mathfrak{d} = \max\{N/d, dB\}.$$

Hence, one would expect the run time to be proportional to

$$(\mathfrak{d}eT)^2.$$

In the experiments below we selected field sizes $q = 3^n$ which arise from supersingular elliptic curves with group orders which are “almost” prime.

4.1 Field Size 3^{186}

This corresponds to a field size of approximately 295 bits. The rough theoretical estimates, given above, for the various values of $e = 1, 2, 4, 6$ are in the table below. A more careful analysis as in [12] reveals the following estimates, where for the factorbases we take the first $T/2$ primes (resp. $T/2$ prime places) on the rational (resp. algebraic side), in other words the value of B is not used directly.

e	Rough Analysis			Analysis of [12]	
	d	B	$T \approx$	d	$T \approx$
1	4	12	89000	5	85000
2	4	6	180000	4	75000
3	4	4	270000	4	150000
6	3	2	530000	4	330000

We compared these results to the yields provided by our implementation and found that the best possible values seemed to be given by

e	d	$T \approx$
1	5	70000
2	4	90000
3	4	190000

This latter table was produced by comparing the yields of the implementation over a fixed time period for various parameter sizes and then selecting the one which would produce a full matrix in the shortest time period. We were however unable to generate suitable experimental data for the case $e = 6$ since this field is really too large to apply the FFS method in practice for such a value of n .

4.2 Field Size 3^{222}

This corresponds to a field size of approximately 352 bits. The rough theoretical estimates, given above, for the various values of $e = 1, 2, 4, 6$ along with the more precise estimates of [12], are in the following table

e	Rough Analysis			Analysis of [12]	
	d	B	$T \approx$	d	$T \approx$
1	4	13	250000	5	130000
2	4	6	180000	4	190000
3	4	4	270000	4	270000
6	4	2	530000	4	460000

We compared these results to the yields provided by our implementation and found that the best possible values seemed to be given by

e	d	$T \approx$
1	5	100000
2	4	190000
3	4	320000

Again the values for $e = 6$ are not given since n is still too small for this base field size to apply the FFS method successfully.

4.3 Field Size 3^{582}

We present this field since it is the first one which is usable in pairing based systems and which “could” be secure against current computing power on both the elliptic curve and the finite field sides. It corresponds to a bit length of 923 bits. The rough theoretical estimates are given below which should be compared to the the analysis in [12]

e	Rough Analysis			Analysis of [12]	
	d	B	$T \approx$	d	$T \approx$
1	5	21	$9.0 \cdot 10^8$	6	$3.0 \cdot 10^9$
2	5	10	$7.0 \cdot 10^8$	7	$3.0 \cdot 10^9$
3	5	6	$1.0 \cdot 10^8$	7	$2.8 \cdot 10^9$
6	5	3	$2.0 \cdot 10^8$	5	$4.0 \cdot 10^8$

Note that these parameters would imply that such key sizes are currently out of range of the FFS method.

To completely confirm both our theoretical estimates and our partial experiments we ran a few experiments through to the final matrix stage and computation of individual logarithms:

Our experiments were run on a network of around 150 Unix based workstations. The network contained a number of older Sparc 5s and 10s running Solaris, plus a large number of Linux based machines with AMD1600 or Pentium 4 processors. We only used idle cycles of the machines whilst other people were using them, hence during the day we had the equivalent of 50 Pentium 4 machines working flat out. At night this increased to around 100 Pentium 4 machines. The matrix step was run on a Sun Blade 1000 workstation.

In the following table we present the wall clock time t_1 needed to produce the relations, the time t_2 needed to solve the matrix step (divided into the time t'_2

needed to perform the structured Gaussian elimination and the time t_2'' to solve the reduced system). We ran the sieving clients for time t_1 producing a total of R relations on T_a elements of the factorbase. For many examples we did not try to find relations on all the factorbase elements, since finding the relations on the last few elements can take a disproportionate amount of time. The line m denotes the approximate row-size of the resulting (approximately square) matrix after the application of structured Gaussian elimination. Our times for the linear algebra step can be considerably improved we believe, and work is ongoing in this area.

n	186	186	186	222	222
N	186	93	62	222	111
e	1	2	3	1	2
d	5	4	4	5	4
$R(X)$	X	X	X	$X^3 + 2X + 1$	$X^3 + X$
T	70000	80000	140000	100000	160000
R	80045	96365	139376	96956	181675
T_a	69345	76425	137750	95169	158952
m	12634	13218	24746	20719	32148
t_1	8h	7h	30h	48h	50h
t_2'	43m	1h	2h 24m	3h	4h 20m
t_2''	13h	16h	1d 18h	2d 21h	4d 14h

5 Conclusion

We have reported on the first implementation of the function field sieve in characteristic three. We have paid particular attention to the case of finite fields which arise in pairing based cryptosystems. In particular such fields are of a composite nature and we have seen that this provides at best a marginal benefit in allowing one to apply the function field sieve over either \mathbb{F}_3 or \mathbb{F}_{3^2} . We have shown how the exact analysis of [12] is more able to predict the behaviour, and thereby parameter choices, than the naive simple analysis. We have also shown how the key sizes one would use in a simple pairing based system are likely to be secure against current algorithms and computing power.

Finally we hope that our work will encourage others to investigate discrete logarithm algorithms in composite fields of characteristic three, and thereby allow the community to have greater faith in the security of pairing based systems which are based over such fields.

References

1. L. M. Adleman. The function field sieve. In *Algorithmic Number Theory Symposium – ANTS-I*, Springer-Verlag LNCS 877, 108–121, 1994.
2. L. M. Adleman and M. A. Huang. Function field sieve method for discrete logarithms over finite fields. *Inform. and Comput.*, **151**, 5–16, 1999.

3. E. A. Bender and E. R. Canfield. An approximate probabilistic model for structured Gaussian elimination. *J. Algorithms*, **31**, 271–290, 1999.
4. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. In *Advanced in Cryptology – CRYPTO 2001*, Springer-Verlag LNCS 2139, 213–229, 2001.
5. P.S.L.M. Barreto, H.Y. Kim, B. Lynn and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – CRYPTO 2002*, Springer-Verlag LNCS 2442, 354–368, 2002.
6. G. Bertoni, J. Guajardo, S. Kumar, G. Orlando, C. Paar and T. Wollinger. Efficient $\text{GF}(p^m)$ arithmetic architectures for cryptographic applications. In *Topics in Cryptology – CT-RSA* Springer-Verlag LNCS 2612, 158–175, 2003.
7. D. Coppersmith. Evaluating logarithms in $\text{GF}(2^n)$. In *16th ACM Symp. Theory of Computing*, 201–107, 1984.
8. D. Coppersmith. Fast evaluation of logarithms in fields of characteristic two. *IEEE Transactions in Information Theory*, **30**, 587–594, July 1984.
9. S.D. Galbraith. Supersingular curves in cryptography. In *Advances in Cryptography – AsiaCrypt - 2001*, Springer-Verlag LNCS 2248, 495–513, 2001.
10. S.D. Galbraith, K. Harrison and D. Soldera. Implementing the Tate pairing. In *Algorithmic Number Theory Symposium – ANTS-V*, Springer-Verlag LNCS 2369, 324–337, 2002.
11. D. M. Gordon and K. S. McCurley. Massively parallel computation of discrete logarithms. In *Advances in Cryptology – CRYPTO'92*, Springer-Verlag LNCS 740, 312–323, 1993.
12. R. Granger. Estimates for discrete logarithm computations in finite fields of small characteristic. In *Cryptography and Coding*, Springer-Verlag LNCS 2898, 190–206, 2003.
13. K. Harrison, D. Page and N.P. Smart,. Software implementation of finite fields of characteristic three. *LMS Journal of Computation and Mathematics*, **5**, 181–193, 2002.
14. A. Joux and R. Lercier. The function field sieve is quite special. In *Algorithmic Number Theory Symposium – ANTS-V*, Springer-Verlag LNCS 2369, 431–445, 2002.
15. B. A. LaMacchia and A. M. Odlyzko. Solving large sparse linear systems over finite fields. In *Advances in Cryptology – CRYPTO '90* Springer-Verlag LNCS 537, 109–133, 1991.
16. C. Lanczos. Solution of systems of linear equations by minimized iterations. *J. of Research of the National Bureau of Standards*, **49**, 33–53, 1952.
17. R. Matsumoto. Using C_{ab} Curves in the Function Field Sieve. *IEICE Trans. Fundamentals*, **82**, March 1999.
18. A.J. Menezes, T. Okamoto and S. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions in Information Theory*, **39**, 1639–1646, 1993.
19. D. Page and N.P. Smart. Hardware implementation of finite fields of characteristic three. In *Cryptographic Hardware and Embedded Systems – CHES 2002*, Springer-Verlag LNCS 2523, 529–539, 2002.
20. C. Pomerance and J. W. Smith. Reduction of huge, sparse linear systems over finite fields via created catastrophes. *Exper. Math.*, **1**, 89–94, 1992.
21. O. Schirokauer. The special function field sieve. *SIAM Journal on Discrete Mathematics*, **16**, 81–98, 2002.
22. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advanced in Cryptology – CRYPTO - 1984*, Springer-Verlag LNCS 196, 47–53, 1984.

23. V. Shoup. NTL – A Library for Number Theory. <http://www.shoup.net/ntl/>.
24. E. Thomé. Computation of discrete logarithms in $\mathbb{F}_{2^{607}}$. In *Advances in Cryptology – AsiaCrypt 2001*, Springer-Verlag LNCS 2248, 107–124, 2001.
25. L.T. Yang and R.P. Brent. The parallel improved Lanczos method for integer factorization over finite fields for public key cryptosystems. In *International Conference on Parallel Programming Workshops (ICPPW)*, IEEE Press, 106-114, 2001.