Two topics in hyperelliptic cryptography

F. Hess[1], G. Seroussi[2] and N.P. Smart[1]

[1] Computer Science Dept.,
Woodland Road,
Bristol University,
Bristol, BS8 1UB, UK.
{florian, nigel}@cs.bris.ac.uk

[2] Hewlett-Packard Labs,
1501 Page Mill Road,
Palo Alto,
CA. 650-857-1501, USA.
seroussi@hpl.hp.com

Abstract. In this paper we address two important topics in hyperelliptic cryptography. The first is how to construct in a verifiably random manner hyperelliptic curves for use in cryptography in generas two and three. The second topic is how to perform divisor compression in the hyperelliptic case. Hence, in both cases we generalise concepts used in the more familiar elliptic curve case to the hyperelliptic context.

1 Introduction

Elliptic curve cryptography was co-invented in 1985 by V. Miller [13] and N. Koblitz [11]. Cryptography based on elliptic curves is especially attractive due to the supposed difficulty of the discrete logarithm problem in the group of rational points on an elliptic curve. In 1989 Koblitz generalised this concept to hyperelliptic curves [12]. In hyperelliptic cryptography the hard problem on which the security is based is the discrete logarithm problem in the divisor class group of the curve.

Whilst elliptic curve cryptography is starting to become commercially deployed, hyperelliptic cryptography is still at the stage of academic interest. This is mainly due to the greater complexity of the underlying arithmetic and the fact that the protocols have been less standardised. One main problem in the hyperelliptic case, as argued in [16], is that it is currently very hard to generate hyperelliptic curves for use in cryptography which do not have any added extra structure. [1] Another problem is that the supporting algorithms which exist in the elliptic curve case have not been fully developed in the hyperelliptic case. In

[1] There is a new general point counting algorithm by Kedlaya [10] for hyperelliptic curves in small odd characteristic. However, it is believed that this algorithm can be extended to the even characteristic case. At present the authors know of no implementation of this algorithm and so cannot we comment on its practical efficiency.

this paper we generalise two such techniques from the setting of elliptic curve cryptography to the setting of hyperelliptic curves.

In the first we give a method to produce hyperelliptic curves in genus two and three which are generated in a verifiably random manner. In the second we give a method to perform divisor compression.

The first contribution is needed to produce suitable curves in a trusted manner. In elliptic curve cryptography, one way to choose a curve is to generate curves at random until one satisfies the correct security requirements. However, someone else then using the system needs to trust that you did not construct a special curve which has some weakness that only you know about. To overcome this problem various standards bodies, e.g. [1], have proposed that the curve is generated in the following manner:

1. Generate in any manner a 160 bit string, S.
2. Using SHA-1 on this string generate some elliptic curve E in a known deterministic manner.
3. Compute the group order N using either the Schoof-Elkies-Atkin algorithm or one of the extensions to Satoh's algorithm, see [3], [14], [15] and [18].
4. If the curve passes the known security checks then publish the triple

$$(S, E, N),$$

   otherwise return to the first step.

Under the assumption that SHA-1 is a one-way function the above method of curve generation prevents the choice of special elliptic curves with secret weaknesses. An elliptic curve chosen in the above way is said to have been chosen "verifiably at random" since any third party given the triple (S, E, N) can check very quickly that not only is the group order N correct but that the curve could not have been created with a known weakness since it would have been computationally impossible to reverse engineer the value of S which gave E using the above algorithm.

We show how the above algorithm can be used to generate verifiably random hyperelliptic curves in characteristic two for use in cryptography. Our method does not produce random hyperelliptic curves taken from the totality of all hyperelliptic curves but produces hyperelliptic curves which have verifiably been constructed in a random manner from a certain well defined subset of all hyperelliptic curves. In other words it is computationally infeasible for us to have created a special curve with some hidden weakness. However, we stress that since our method produces random hyperelliptic curves from a special family it

---

Just before submitting the final version of this paper to the conference proceedings, Pierrick Gaudry informed us that the AGM method presented at the rump session of EUROCRYPT 2001 can now be used to compute the group order of a Jacobian of a hyperelliptic curve in genus two over a field of characteristic two. Indeed the AGM method is practical for cryptographically sized Jacobians. Hence, the AGM method for genus two should therefore be preferred to ours since it allows a truly random curve to be used rather than one from a special family.

is possible that the curves constructed by our method have a weakness which we are not aware of. For further details of how special the families we construct actually are the reader should consult the paper [6].

Previous attempts at generating cryptographically strong hyperelliptic curves have been based on analogues from the elliptic case, namely generalisations of the SEA algorithm or the CM method. In [8] a first attempt at an analogue of the SEA algorithm for hyperelliptic curves of genus two is reported on. The authors manage to compute the order of a random hyperelliptic curve of genus two of group order roughly 2^{126}. However, this takes them many days of computing time. In practice one would need to repeat their method a large number of times before a suitable curve for use in cryptography was determined. Whilst the method in [8] is to be preferred over ours, it can only be used when (and if) the algorithms become sufficiently fast. Our method on the other hand, as we have already stated, is practical using today's knowledge and technology.

A number of authors have looked at using an analogue of the CM method to generate hyperelliptic curves for use in cryptography, [17], [19] and [5]. However, this has a number of draw backs compared to our method above. Firstly, the existing literature on applying the CM method to hyperelliptic curves only applies to large odd characteristic and not characteristic two as our method does. Secondly, the set of curves produced by the CM method in practice, if one could implement it in characteristic two, would be from a far more restricted set than the set of curves generated by our method.

Our second contribution is to give a method in all characteristics to perform divisor compression. In the elliptic curve case it is common practice to use a technique called point compression to reduce the sizes of the public keys being transported by fifty percent. This is done by noticing that an elliptic curve point (x, y) can be represented by x and a bit to decide which value of y to use. This is particularly important when deploying ECC in an environment where bandwidth is constrained. We will show that the elliptic curve point compression techniques can be naturally generalised to the hyperelliptic setting.

The first author would like to thank J. Cannon for his support while this work was in preparation.

2 Producing Hyperelliptic Curves

Our technique of producing hyperelliptic curves verifiably at random is based on the method of Weil restriction of scalars as outlined in [9]. In this technique one takes an elliptic curve E over the field $K = \mathbb{F}_{q^n}$, where q is a power of two and then one constructs a hyperelliptic curve H over the subfield $k = \mathbb{F}_q$. Since the groups $E(K)$ and $\mathrm{Jac}_k(H)$ are related by a group homomorphism one can easily compute, in certain cases, the group order of $\mathrm{Jac}_k(H)$.

To fix notation we are trying to generate a hyperelliptic curve H over the field $\mathbb{F}_q$, of genus g and of group order $N = 2^l p$, where p is a prime. Before giving our technique for the generation of hyperelliptic curves we need to summarise the main security requirements for our curve.

- $p > 2^{160}$. This is to protect against Pohlig-Hellman, Pollard-rho and Baby-Step/Giant-Step attacks.
- $g < 4$. This is to protect against the method of Gaudry [7].
- $q = 2^r$, where r is prime. This is to protect against using Weil descent on $\mathrm{Jac}_{\mathbb{F}_q}(H)$.
- The smallest $s \geq 1$ such that $q^s \equiv 1 \pmod{p}$ should be greater than $20g$. This is to protect against the Tate-pairing attack [4].

Note, there are no other conditions which give curves with a known weakness and all the above conditions can be easily checked given the curve and its group order.

In [9] a method is given for finding a group homomorphism from an elliptic curve defined over $\mathbb{F}_{q^n}$ to a hyperelliptic curve H defined over $\mathbb{F}_q$. The technique given is completely deterministic, although the resulting model for H is not in the standard form, an issue which we shall return to below. The method of [9] uses a set of Artin-Schreier extensions, the number of distinct extensions being given by an integer m, which satisfies $1 \leq m \leq n$. For the exact definition of m see [9], all that we shall require is that $m = n$ and that the genus of the resulting hyperelliptic curve is either 2^{m-1} or $2^{m-1} - 1$. In our applications we are able to control precisely when we obtain genus 2^{m-1} or genus $2^{m-1} - 1$.

Since we wish to produce hyperelliptic curves with Jacobians of the same group order as $E(K)$ we need to choose elliptic curves so that

$$n = 2^{m-1} \text{ or } n = 2^{m-1} - 1.$$

Since one of our security requirements on g is that it should be less than four, these conditions are easy to satisfy.

For cryptographic purposes it is advantageous to produce a model for the hyperelliptic curve of the form

$$H : Y^2 + H(X)Y = F(X)$$

where $\deg H(X) \leq g$ and $\deg F(X) = 2g + 1$. Such a model will be called "reduced" and we shall now describe a deterministic method to turn the hyperelliptic model, produced by the method of [9], into a reduced model. This is important, and was not addressed in [9]. If we wish to generate hyperelliptic curves verifiably at random we require a deterministic mapping from the elliptic curve to a reduced model of a hyperelliptic curve.

Assume that a fixed representation has been chosen for the finite fields of size q^n and q. Using this fixed representations we can define (lexicographical) orders in the finite fields, hence orders on polynomials, matrices etc. Utilising normalisation of polynomials, polynomial division, Hermite normal forms and other such reduction techniques we are then able to always consider the smallest (or the same) object having a desired property.

Taking the model for H produced by the method in [9] we then move the smallest rational point to infinity. A reduced hyperelliptic equation is then obtained by computing the minimal polynomial over the rational subfield of a function of smallest odd pole order at infinity and with no other poles.

Since the algorithm, outlined above, to proceed from an elliptic curve to a reduced model for a hyperelliptic curve is completely deterministic, all we need do to produce a verifiably "random" hyperelliptic curve is to find an elliptic curve verifiably at "random" with the required properties.

2.1 Genus Two

Take a finite field of the form $K = \mathbb{F}_{q^2}$ where q is 2 raised to a prime exponent. We construct, using the technique from [1] a verifiably random elliptic curve of the form

$$Y^2 + XY = X^3 + aX^2 + b$$

where $a, b \in K$, with group order equal to $2p$ where p is a prime number. Note that since p is a prime number and q is 'large', in the Weil descent we almost always obtain $m = 2$ and so the resulting hyperelliptic curve will have genus two. Then using the technique of Weil descent we can construct a hyperelliptic curve over the field $k = \mathbb{F}_q$ which has group order divisible by p. Since the Weil restriction of E and $\mathrm{Jac}_k(H)$ have the same dimension, they are therefore isogenous. But they then have the same number of points over k and so $\mathrm{Jac}_k(H)$ will have group order exactly $2p$.

2.2 Genus Three

For genus three we need to proceed in a slightly different way. First we choose a finite field of the form $K = \mathbb{F}_{q^3}$ where again q is 2 raised to a prime exponent. Then we take an random 160-bit string and pass it through SHA-1 to obtain a field element $v \in \mathbb{F}_{q^3}$ using the methods of [1]. Setting $b = v + v^q$ we see that

$$\mathrm{Tr}_{K/k}(b) = 0.$$

We then compute the elliptic curve

$$Y^2 + XY = X^3 + X^2 + b$$

and its group order. This is repeated until we find a group order equal to $2p$ where p is a prime. Then using the arguments of [9] we will obtain a hyperelliptic curve of genus three. Although we are not choosing elliptic curves completely at random from all elliptic curves defined over K, we are choosing them uniformly at random from a subset of size q^2. Just as before, we will have that $\mathrm{Jac}_k(H)$ has group order exactly $2p$.

Our technique for constructing hyperelliptic curves for use in cryptography is dominated by the time needed to apply the Schoof-Elkies-Atkin (SEA) algorithm or the algorithm of Satoh to a set of elliptic curves, until one with the correct cryptographic properties is determined. The step of transforming the elliptic curve into a hyperelliptic curve only takes a few seconds. Hence, to compute a

single hyperelliptic curve of genus two with the correct cryptographic properties takes, for a Jacobian of size roughly 2^{190}, on the order of a couple of minutes. The main computational task is to repeatedly apply the SEA/Satoh algorithm until a suitable elliptic curve is found. Of course, exact times depend strongly on the details of the SEA/Satoh implementation

Finally to end this section we give a typical example:
n=166
Elliptic Curve : K is defined by $w^{166} + w^{37} + 1 = 0$

$$\begin{aligned} S &= \text{E4D1C989A8999ED0EF8AC7D691E5D8ADDAD481F5}, \\ a &= \text{3951AD54028E7E3CF2D437A4186CCB53BF5DD39196}, \\ b &= \text{140463F3747C98BAE9D9D31EAF3FCE65ADF80AEA26}, \\ N &= \text{3FFFFFFFFFFFFFFFFFFF730032E01F3184452AA1A}. \end{aligned}$$

Hyperelliptic Curve : k is defined by $t^{83} + t^7 + t^4 + t^2 + 1 = 0$.

$$\begin{aligned} H(X) = {} & \text{6C935CFDD963AD086B738}X^2 + \text{103FEA81D67CBF0210A96}X \\ & + \text{47242588808C36BFBE701}, \\ F(X) = {} & \text{660212F23F5C16AE899A9}X^5 + \text{6CAEC90C545CF269FE5B1}X^4 \\ & + \text{5A55B3786562759A427E0}X^3 + \text{32C4479705A4CEBF1FEA3}X^2 \\ & + \text{7F018AAEC622917758194}X + \text{2BDCB9CD696E5142054C8}. \end{aligned}$$

3 Divisor Compression

As noted previously point compression in the elliptic curve case is an important tool used to save around fifty percent of the bandwidth in transferring/storing public keys and in Diffie-Hellman key exchange. Before describing our analogous method in the hyperelliptic setting we shall describe the exact data format normally used for divisors on hyperelliptic curves. For more details on what follows the reader should consult the papers by Cantor [2] and Koblitz [12]. In this section we shall work with arbitrary characteristic fields.

A hyperelliptic curve of genus g, over a field k of characteristic p, we will assume is given by an equation of the form

$$Y^2 + H(X)Y = F(X),$$

where $H(X), F(X) \in k[X]$, $\deg H(X) \leq g$ and $\deg F(X) = 2g + 1$. For applications it is common to assume that either p is very large or equal to two. If p is large we usually assume that $H(X) = 0$. Notice that in characteristic two the ramified places lying above $p(X) \in k[X]$ are exactly those for which $p(X)$ divides $H(X)$.

The group elements, upon which our cryptographic protocols operate, are effective reduced divisors of degree less than or equal to g. Such a divisor can be represented by the pair

$$D = (a(X), b(X)),$$

where $a(X), b(X) \in k[X]$, $\deg b(X) < \deg a(X) \leq g$, $a(X)$ is monic and

$$b(X)^2 + H(X)b(X) - F(X) \equiv 0 \pmod{a(X)}.$$

The zero in the group is represented by the pair $(1, 0)$. That the divisor is reduced means that no ramified place occurs in the support of D with multiplicity greater than one, and that if a place $\mathfrak{p}$ occurs in the support of D then the image of $\mathfrak{p}$ under the hyperelliptic involution does not. In many protocols one needs to transmit divisors, naively this requires at most g elements of k to represent $a(X)$ and at most g elements of k to represent $b(X)$.

However, given $a(X)$ there are only a small number of possible values for $b(X)$ which could correspond to $a(X)$. We shall show how one can recover the correct $b(X)$ from only $a(X)$, and at most an additional g bits of information.

Our first task is to decide a canonical order on the irreducible polynomials of degree less than or equal to g, which are defined over k. This is done by fixing a field representation and using the lexicographic order used for a similar purpose in Section 2.

When we are either compressing or decompressing we first factorize $a(X)$ into its irreducible factors and order them. Since factorisation of polynomials can be performed in random polynomial time, and in applications the degree of $a(X)$ will be quite small (usually less than four) this factorisation stage is no barrier to our method.

For example when $g = 2$ we need to factorize a degree two polynomial. This factors either when a certain trace is zero, for the even characteristic case, or when the discriminant is a square, for the odd characteristic case. In either characteristic we can easily deduce the factorisation when the polynomial is reducible using standard techniques for solving quadratic equations over finite fields. Similar considerations apply when $g = 3$.

Each irreducible factor $p(X)$ of $a(X)$ will correspond to at most two prime divisors on H:

$$D_p = (p(X), q(X)) \text{ and } D'_p = (p(X), -q(X) - H(X) \pmod{p(X)}),$$

where $q(X)$ is the polynomial of least degree such that

$$q(X)^2 + H(X)q(X) - F(X)$$

is divisible by p. Since the divisor we are compressing or decompressing is reduced we know that only one of these two possibilities is in the support of D. Hence, for each prime divisor of $a(X)$ we need only specify one bit of information to determine whether D_p or D'_p is in the support of D. The only questions remaining are how to produce this bit and how to recover the correct value of $b(X)$, given $a(X)$ and the resulting bits.

3.1 Compression

The basic idea is to execute the following steps for every distinct irreducible factor $p(X)$ of $a(X)$, this gives the bits β_p.

1. If $p(X)$ is ramified in $k(H)$ set $\beta_p = 0$.
2. If the characteristic of k is odd, and so $H(X) = 0$, then let β_p denote the parity of the smallest non-zero coefficient of $b(X) \pmod{p(X)}$.
3. If the characteristic of k is even then we set

$$\overline{t}(X) = b(X)/H(X) \pmod{p(X)},$$

notice that the inversion of $H(X)$ modulo $p(X)$ can be accomplished since $p(X)$ is unramified and so $\gcd(p(X), H(X)) = 1$, We then let β_p denote the least significant bit of the constant term of $\overline{t}(X)$.

Hence, the compressed form of the divisor D is $\{a(X), s\}$ where s is the bit string containing the β_p for each irreducible factor of $a(X)$. The bit string is ordered with respect to the ordering on the distinct irreducible factors of $a(X)$.

3.2 Decompression

Suppose $p(X)^k$ exactly divides $a(X)$, then if we can recover $b(X)$ modulo $p(X)^k$ for all irreducible factors $p(X)$ of $a(X)$ we can then recover $b(X)$ either via the Chinese Remainder Theorem or by adding together the local components for each prime $p(X)$.

Since $(a(X), b(X))$ is a reduced divisor, we know that if $p(X)$ is ramified then the value of k above is one, and recovering $b(X)$ modulo $p(X)$ is trivial, since it will be equal to zero modulo $p(X)$.

We now turn to the case where $p(X)$ is not ramified. Then recovering $b(X)$ modulo $p(X)^k$, is trivially done once we know $b(X) \pmod{p(X)}$. This recovery of $b(X)$ modulo $p(X)^k$ from $b(X) \pmod{p(X)}$ can be accomplished in one of two ways:

1. Using Hensel's Lemma.
2. By multiplying the divisor $(p(X), b(X) \pmod{p(X)})$ by k.

So we have reduced the decompression problem to determining the value of

$$b(X) \pmod{p(X)}$$

given $p(X)$ and the bit β_p.

Since $p(X)$ is irreducible, the algebra $k[X]/p(X)$ is a field and we can apply well known techniques to solve quadratic equations in a field to determine a candidate value $\overline{b}(X)$ for $b(X) \pmod{p(X)}$. To check whether $\overline{b}(X)$ is the correct value we compute the value of the bit β_p, as in the compression algorithm, assuming that $\overline{b}(X)$ is correct. If this value agrees with the supplied value then we know that $\overline{b}(X) = b(X) \pmod{p(X)}$, otherwise we set $b(X) = -\overline{b}(X) - H(X) \pmod{a(X)}$.

Finally, note that the above algorithms for divisor compression and decompression are only slightly more complicated than those used in the elliptic curve case.

References

1. X9.62 Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA). *American National Standards Institute*, 1999.
2. D.G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, **48**, 95–101, 1987.
3. M. Fouquet, P. Gaudry and R. Harley. An extension of Satoh's algorithm and its implementation. *J. Ramanujan Math. Soc.*, **15**, 281–318, 2000.
4. G. Frey and H.-G. Rück. A remark concerning m-divisibility and the discrete logarithm problem in the divisor class group of curves. *Math. Comp.*, **62**, 865–874, 1994.
5. G. Frey, Applications of arithmetical geometry to cryptographic constructions, Preprint, 2000.
6. S.D. Galbraith. Limitations of constructive Weil descent. To appear *Proceedings of a Conference on Cryptography and Computational Number Theory, Warsaw, Sept 2000.*
7. P. Gaudry. An algorithm for solving the discrete log problem on hyperelliptic curves. *Advances in Cryptology, EUROCRYPT 2000*, Springer-Verlag LNCS 1807, 19–34, 2000.
8. P. Gaudry and R. Harley. Counting points on hyperelliptic curves over finite fields. *ANTS-IV*, Springer-Verlag LNCS 1838, 313–332, 2000.
9. P. Gaudry, F. Hess and N.P. Smart. Constructive and destructive facets of Weil descent on elliptic curves. To appear *J. Cryptology.*
10. K. Kedlaya. Counting points on hyperelliptic curves using Monsky-Washnitzer cohomology. Preprint 2001.
11. N. Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, **48**, 203–209, 1987.
12. N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, **1**, 139–150, 1989.
13. V. Miller. Use of elliptic curves in cryptography. *Advances in Cryptology, CRYPTO - '85*, Springer-Verlag LNCS 218, 47–426, 1986.
14. T. Satoh. The canonical lift of an ordinary elliptic curve over a finite field and its point counting. *J. Ramanujan Math. Soc.*, **15**, 247–270, 2000.
15. B. Skjernaa. Satoh's algorithm in characteristic two. Preprint 2000.
16. N.P. Smart. On the performance of hyperelliptic cryptosystems. *Advances in Cryptology, EUROCRYPT '99*, Springer-Verlag, LNCS 1592, 165–175, 1999.
17. A.-M. Spallek, *Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen*, PhD Thesis, IEM Essen, 1994.
18. F. Vercauteren, B. Preneel and J. Vandewalle. A memory efficient version of Satoh's algorithm. *Advances in Cryptology, EUROCRYPT 2001*, Springer-Verlag, LNCS 2045, 1–13, 2001.
19. A. Weng, Constructing hyperelliptic curves of genus 2 suitable for cryptography, Preprint, 2000.