Point Compression on Jacobians of Hyperelliptic Curves over $\mathbb{F}_q$.

Colin Stahlke

ABSTRACT. — Hyperelliptic curve cryptography recently received a lot of attention, especially for constrained environments. Since there space is critical, compression techniques are interesting. In this note we propose a new method which avoids factoring the first representing polynomial. In the case of genus two the cost for decompression is, essentially, computing two square roots in $\mathbb{F}_q$, the cost for compression is much less.

Introduction

An asymmetric cryptographic system such as ElGamal's needs a finite group such as the Jacobian of a hyperelliptic curve over a finite field. For genus $g = 1$ we have the well-known special case of elliptic curve cryptography.

Since these cryptographic systems are realized in computers with limited ressources (eg. smart cards) and communication happens over channels with limited band width, it is desirable that the representation of a group element need little space.

How much space in bits does a point on a Jacobian over $\mathbb{F}_q$ need? Its representation as a tuple of g points on the curve with x and y coordinates needs $2g \cdot \log_2(q)$ bits. On the other hand, since the number of points on the Jacobian is close to q^g, the amount of information in a point is only about $g \cdot \log_2(q)$ bits.

In a cryptographic context every point of interest is in the cyclic group generated by some known point P of the Jacobian. So an optimal point compression would be: Given a point Q, calculate $k \in \mathbb{Z}$ such that $0 \leq k < \#\langle P \rangle$ and $Q = k \cdot P$. Since $\#\langle P \rangle$ is at most the number of points on the Jacobian and k identifies the point Q uniquely, k is the optimal compression of the point Q in the sense that k needs exactly as many bits as there is information in the choice of the point Q. Obviously this compression is not practical since compressing a point would be calculating its discrete logarithm which means breaking the crypto system. So we need to find a trade off between a good compression and the computing power needed to compress and decompress a point.

Fast point addition usually uses the Mumford representation. Lange found the fastest formulas so far (see [Lange]). In the Mumford representation each element of the Jacobian is represented by a pair of polynomials $[u(x), v(x)]$ of bounded degree. Hess, Seroussi, and Smart [HSS] propose a method for compression where each element is represented by at most $g + g \log_2 q$ bits. In this note we propose a different technique needing the same amount of space but the computing costs are lower.

1. The Mumford representation

In [Mum][page 3.17] Mumford introduces the following representation of ideal classes which correspond to divisor classes, i.e. to points on the Jacobian:

Theorem (Mumford Representation): Let the function field be given via the absolutely irreducible polynomial $y^2 + h(x)y = f(x)$, where $h, f \in \mathbb{F}_q[x]$, $\deg f = 2g+1$, $\deg h \leq g$. Each nontrivial ideal class over $\mathbb{F}_q$ can be represented via a unique ideal generated by $u(x)$ and $y - v(x)$, $u, v \in \mathbb{F}_q[x]$, where

- u is monic,
- $\deg v < \deg u \leq g$,
- $u | v^2 + vh - f$.

Let $D = \sum_{i=1}^{r} P_i - r\infty$, where $P_i \neq \infty$, $P_i \neq \iota P_j$ for $i \neq j$ and $r \leq g$ (ι is the hyperelliptic involution). Put $P_i = (a_i, b_i)$. Then the corresponding ideal class is represented by $u = \prod_{i=1}^{r}(x - a_i)$ and if P_i occurs n_i times then $\left(\frac{d}{dt}\right)^j \left[v(x)^2 + v(x)h(x) - f(x)\right]_{|x=a_i} = 0$, $0 \leq j \leq n_i - 1$.

Now we want to compress a representative $[u, v]$ of an ideal class by storing u and some more bits, such that we can reconstruct v.

Since $u | v^2 + vh - f$, there is a $p \in \mathbb{F}_q[x]$ such that $up = v^2 + vh - f$. This is an equation between two polynomials of degree $2g+1$ since $\deg f = 2g+1$. The unknowns are the $2g + 2 - \deg u$ coefficients of p and at most $\deg u$ coefficients of v. Therefore by comparision of coefficients we have $2g+2$ equations with at most $2g+2$ unknowns. We expect at most 2^g solutions for v in which case the choice of one solution can be encoded in g bits.

This expectation has to be verified in each case of course. For elliptic curves we get 2 solutions for v and with just one bit we can reconstruct v which corresponds to calculating the y-coordinate from the x-coordinate of a point. By way of illustration from now on we restrict ourselves to genus $g = 2$ and odd characteristic. Then a curve can be defined by

$$y^2 = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0,$$

where f has only simple roots in the algebraic closure. Let f_i be the coefficients of f, u_i of u, v_i of v and p_i of p. From $up + f = v^2$ by comparision of coefficients we get

$$p(x) = -x^3 + (u_1 - f_4)x^2 + (u_0 - u_1^2 + f_4u_1 - f_3)x + p_0.$$

The discriminant of

$$\begin{aligned}(1)\quad u(x)p(x) + f(x) = {} & (p_0 + f_2 - f_3u_1 - f_4(u_0 - u_1^2) + u_1(2u_0 - u_1^2))\, x^2 + \\ & (u_1p_0 + f_1 - f_3u_0 + f_4u_0u_1 + u_0(u_0 - u_1^2))\, x + \\ & u_0p_0 + f_0\end{aligned}$$

is of degree at most 2 in p_0 and all coefficients are known. It is zero since $u(x)p(x) + f(x)$ is a square (namely $v(x)^2$). From $v(x)^2 = u(x)p(x) + f(x)$ we get relations for v_0 and v_1:

$$\begin{aligned}(2)\quad v_0^2 &= u_0p_0 + f_0 \\ (3)\quad 2v_0v_1 &= u_1p_0 + f_1 - f_3u_0 + f_4u_0u_1 + u_0(u_0 - u_1^2) \\ (4)\quad v_1^2 &= p_0 + f_2 - f_3u_1 - f_4(u_0 - u_1^2) + u_1(2u_0 - u_1^2)\end{aligned}$$

2. Compression

The f_i, u_i and v_i are known. Calculate p_0 from (2) or (if $u_0 = 0$) from (4). Consider the right hand side of (1) as polynomial in x and let $d(p_0)$ be its discriminant which we consider as polynomial in p_0. If $u_1^2 - 4u_0 \neq 0$, consider the discriminant of d and decide which root gives the correct value for p_0. Store this choice in Bit_1. Since q is odd, the most convenient choice might be to take as Bit_1 the least significant bit of the root (i.e. the parity of a coordinate of the root considered as number in $[0, p-1]$).

Exception: If $u_1^2 - 4u_0 = 0$ then $d(p_0)$ is of degree 1 and Bit_1 can be chosen arbitrarily. In fact $d(p_0)$ is never of degree 0, otherwise a short calculation shows that $f(-u_1/2) = 0$ and $f'(-u_1/2) = 0$, so $-u_1/2$ would be a singular point of the curve.

Now store in Bit_2 the correct choice of v_0 as root of $u_0p_0 + f_0$ (see (2)). (Again the most convenient choice might be to take as Bit_2 the least significant bit of v_0). But if $v_0 = 0$ then instead store in Bit_2 the correct choice of v_1 as root of the right hand side of (4).

The compressed point is the tuple (u_0, u_1, Bit_1, Bit_2).

3. Decompression

The f_i and u_i are known. Also known are Bit_1 and Bit_2. We need to recover v_0 and v_1. Take the discriminant of $u(x)p(x) + f(x)$ (see (1)) and consider this discriminant $d(p_0)$ as polynomial in p_0. Calculate p_0 from $d(p_0) = 0$ according to Bit_1. Calculate v_0 from (2) according to Bit_2. If $v_0 \neq 0$ then calculate v_1 from (3). If $v_0 = 0$ then calculate v_1 from (4) according to Bit_2.

References

[HSS] F. Hess, G. Seroussi, and N. P. Smart. Two topics in hyperelliptic cryptography. In *Selected Areas in Cryptography – SAC 2001*, volume 2259 of *Lect. Notes Comput. Sci.*, pages 181–189. Springer, 2001.

[Lange] T. Lange. Formulae for Arithmetic on Genus 2 Hyperelliptic Curves. `http://www.itsc.ruhr-uni-bochum.de/tanja/preprints.html`, 2003. submitted.

[Mum] D. Mumford. *Tata Lectures on Theta II.* Birkhäuser, 1984.

EDIZONE GmbH, Siegfried-Leopold-Str. 58, D-53225 Bonn, Germany
e-mail: stahlke@edizone.de