CHES 2002, San Francisco

On the Efficient Generation of Elliptic Curves Over Prime Fields

E. Konstantinou, Y.C. Stamatiou,and C. Zaroliagis

Department of Computer Engineering and Informatics, University of Patras, Greece &

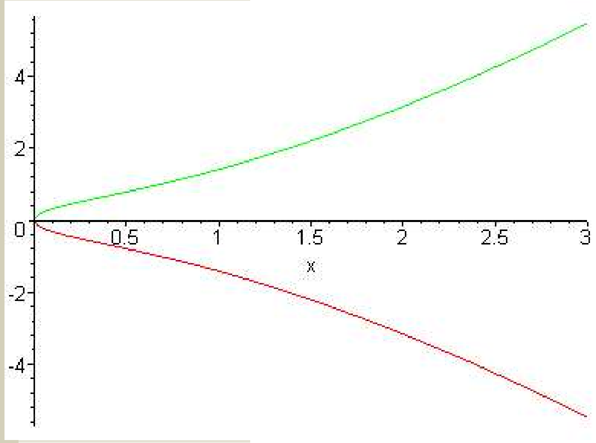
Computer Technology Institute, Patras, Greece

Elliptic Curves

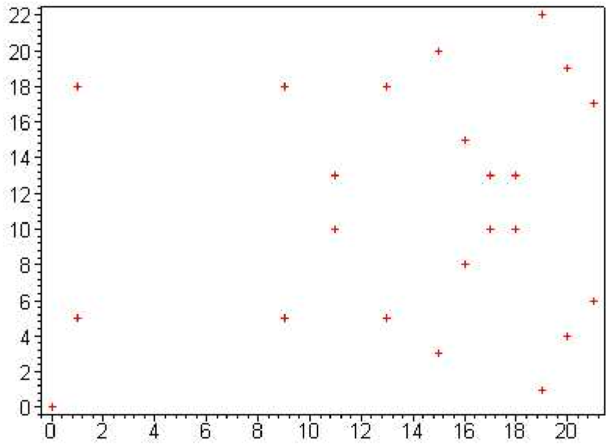
- The set of solutions (x,y) to

$$y^2 = x^3 + ax + b$$

- Usually defined over a prime or binary field.
- E.g., $y^2 = x^3 + x$:



Q

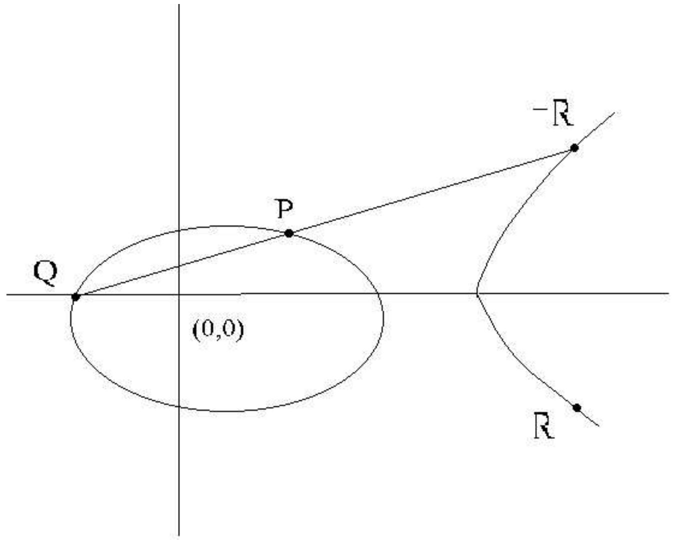


F_{23}

Basic EC Algebraic Operations

(Point addition)

- R = P + Q



(Scalar multiplication by an integer)

- $Q = kP = \underbrace{P + \ldots + P}_{k \text{ times}}$

Generating Elliptic Curves

Three methods:

- ❑ **Constructive Weil descent**
  - ➞ Samples from a, rather, limited subset of ECs.
- ❑ **Point counting** (Based on *Schoof's point counting method*)
  - ➞ Rather slow
- ❑ The **Complex Multiplication** method
  - ➞ Rather involved implementation, but more efficient and guarantees construction of ECs of crypto strength.

Properties of Secure ECs

- To ensure intractability of the ECDLP by all known attacks, the EC group order, *m*, should satisfy the following conditions:

✓ $\boldsymbol{m = nq}$ where q a prime $> 2^{160}$

$\longrightarrow$ Avoids Pohlig-Hellman, Pollard-Rho attacks

✓ $\boldsymbol{m \neq p}$ (the order of F_p)

$\longrightarrow$ Avoids *anomalous attack*

✓ $\boldsymbol{p^k \neq 1 \pmod{m}}$ for all $1 \leq k \leq 20$

$\longrightarrow$ MOV attack

The general CM Method

- **Objective:** Build an EC of ***prescribed*** order having the security properties shown before.
- **Method:**

- ***Given*** prime p, find the smallest D so that $4p = u^2 + Dv^2$.
- Check whether either $\mathbf{m = p + 1 - u}$ or $\mathbf{m = p + 1 + u}$ has the security properties.
- Construct the ***Hilbert polynomial*** corresponding to D.
- Find a root modulo p of the polynomial.
- Construct the ECs with the root as invariant.
- Choose the curve having the order determined in previous step.

Shortcomings of the CM method

- Time consuming construction of Hilbert polynomials (required precision, root location etc.) as *D* increases – huge polynomial coefficients
- Each time a new prime is constructed, a *D* is selected that was possibly used before with some other prime – construction of the same polynomials
- Need for improvements, especially for hardware devices where ***memory*** and ***speed*** are limited resources

Improvement!

Savaş, Schmidt, Koç, *CHES* 2001:

- As Hilbert polynomials depend only on *D*, precompute Hilbert polynomials for a specific set of *D* values
- Then choose a *D* from among this set, avoiding recomputation of the polynomials
- **For various u, v test whether $p = (u^2 + Dv^2)/4$ is prime**
- Determine the curve order as before
- Finally, locate the roots (this depends on *p*) and construct the appropriate elliptic curve

Possible problem: large memory requirements for storing Hilbert polynomials

Our approach

- Basically the usual CM method
- On line computation (or precomputation) of ***Weber polynomials***
- Roots of these polynomials are easily ***transformed*** into the roots of the corresponding Hilbert polynomials but no Hilbert polynomial is actually constructed
- **But why use Weber polynomials?**

Weber vs. Hilbert Polynomials

- The construction of both types of polynomials requires high precision complex, floating point arithmetic.
- ***Drawback*** of Hilbert polynomials: their fast growing (with *D*) coefficients - time consuming construction and difficult to implement in limited resources devices.
- Weber polynomials on the other hand, have ***much smaller*** coefficients.

An Example (D = 292):

$$W_{292}(x) = x^4 - 5x^3 - 10x^2 - 5x + 1$$

$$H_{292}(x) = x^4 - 206287709804283046 08 \cdot 10^2\, x^3 - 93693622511929038759497066112 \cdot 10^6\, x^2 + 45521551386379385369629968384 \cdot 10^9\, x - 380259461042512404779990642688 \cdot 10^{12}$$

(A lot of trailing zeros!)

The Details of our CM Variant

Preprocessing Phase:

1. Choose a discriminant *D*.
2. **Construct the Weber (or Hilbert) polynomial (on-line or off-line).**

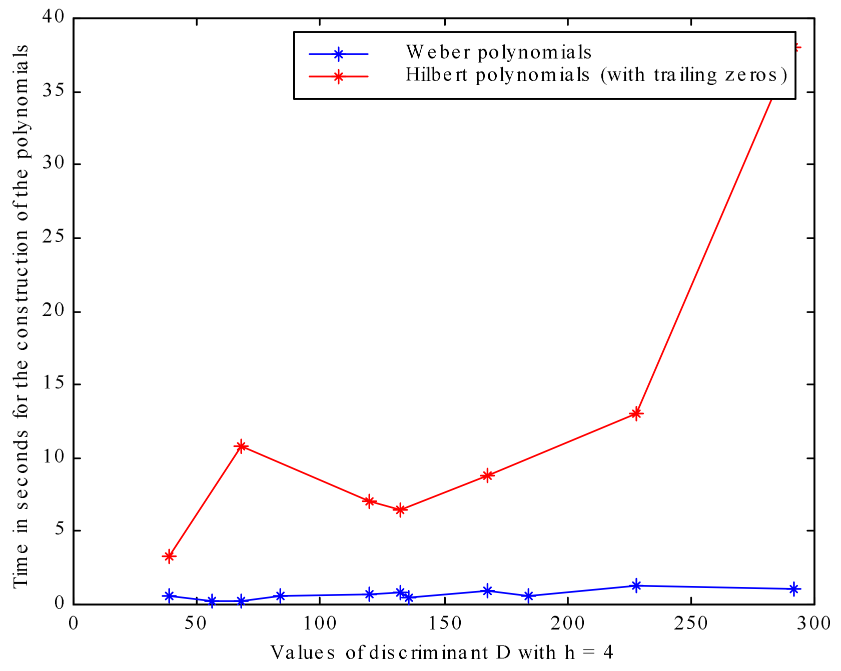
Main Phase:

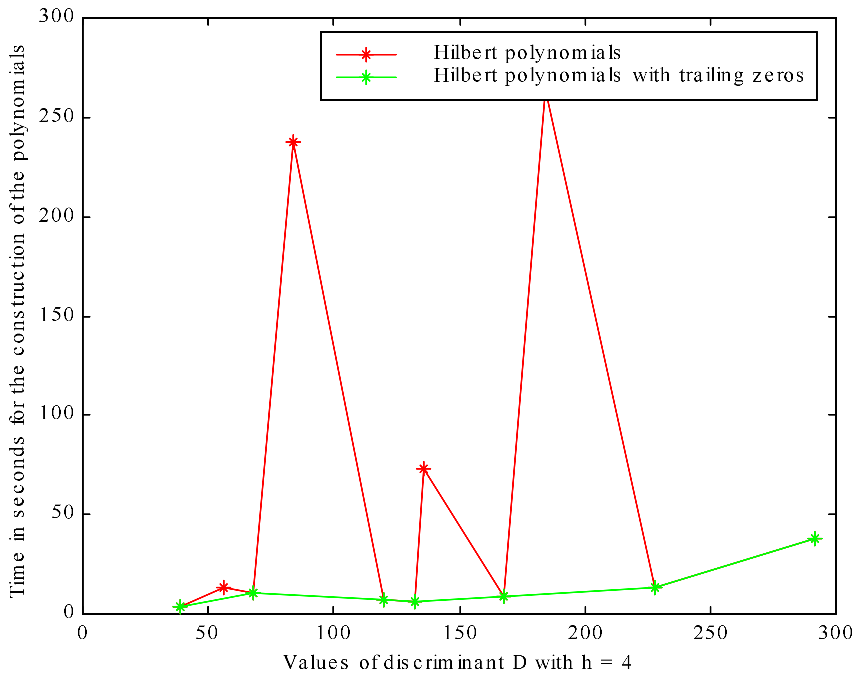
1. **Produce a random prime *p*** and check if there are integers (u,v) satisfying $4p = u^2 + Dv^2$ (using Cornacchia's algorithm). If not, repeat.
2. Possible curve orders: $\underline{\boldsymbol{m = p + 1 - u}}$ and $\underline{\boldsymbol{m = p + 1 + u}}$. Check if at least one of them is ***suitable***. If not, return to the previous step.
3. Compute the roots of the polynomial modulo *p*. Transform roots of Weber polynomial (if Weber polynomials were chosen) to roots of the corresponding Hilbert polynomial.
4. Each root represents a *j*-invariant, leading to two elliptic curves.
5. Choose the curve which has order *m* (probabilistic check).

Implementation Environment

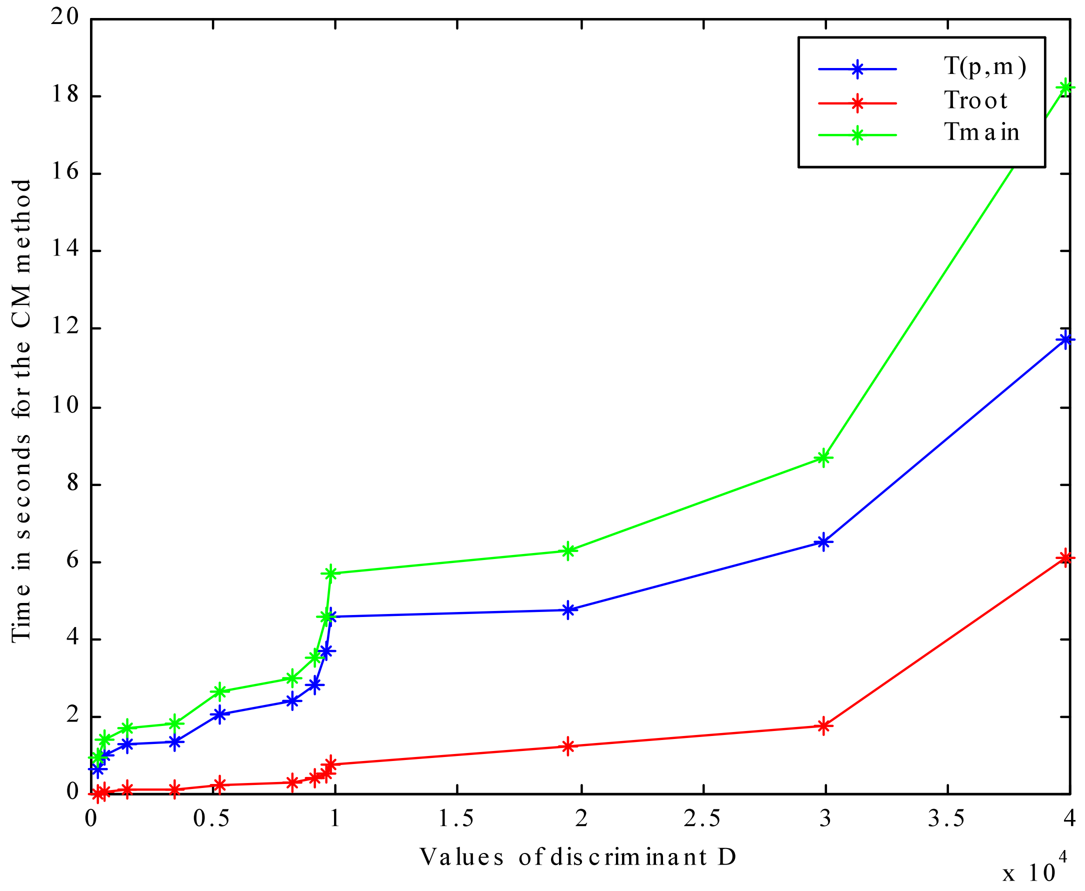
- The experiments were carried out on a Pentium III (933 MHz) with 256 MB of main memory, running SuSE-Linux 7.1, using the ANSI C gcc-2.95.2 compiler with the GNUMP library.
- **Code size:** 69Kbytes, including the code for the polynomials, or 56Kbytes without this code.

Running Times (Polynomials)

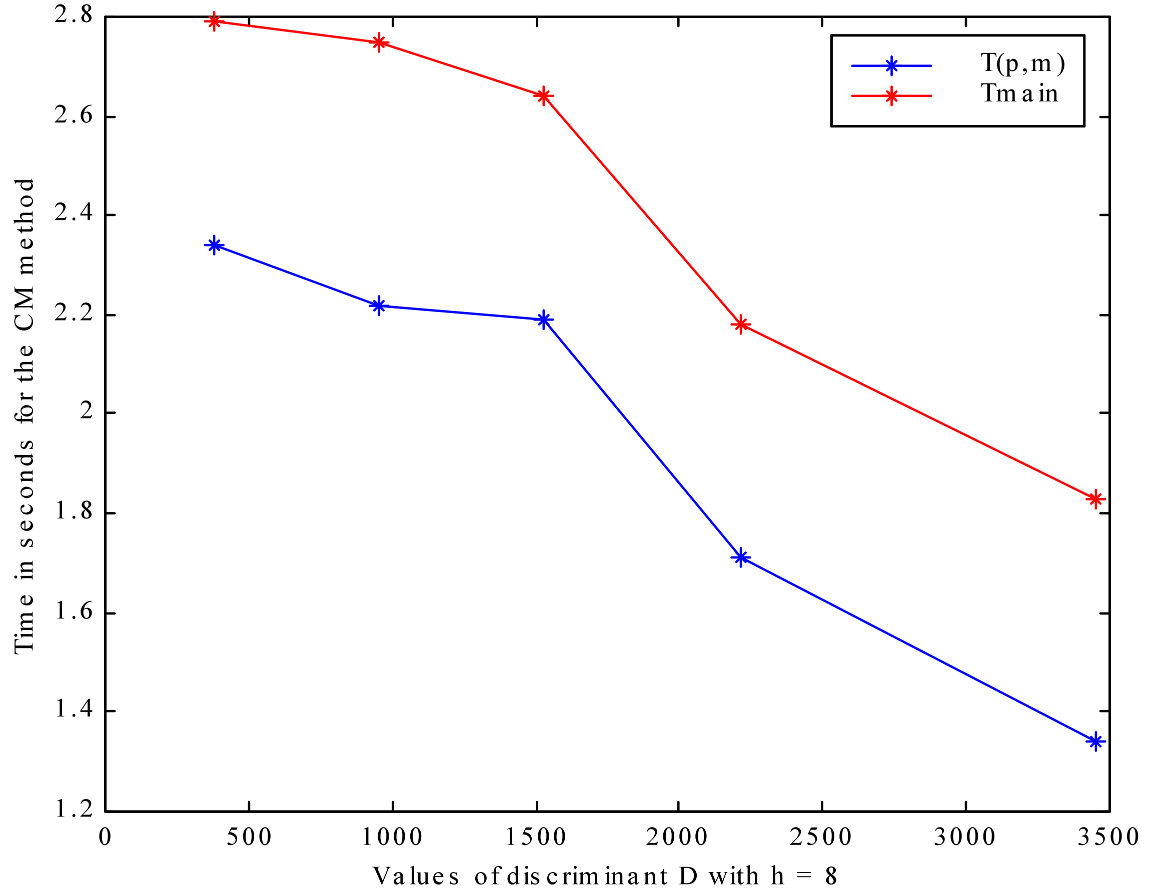




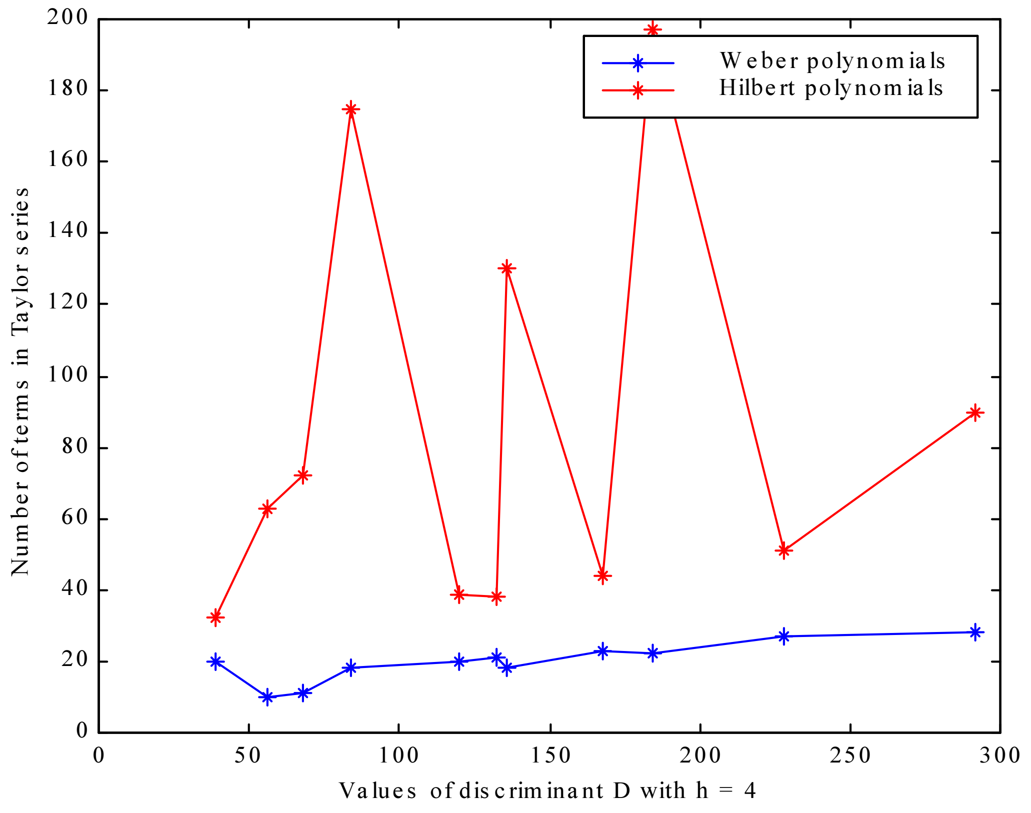
Running Times (our CM Variant)



The case *h* = 8 (our CM Variant)



Required Precision (Taylor Series Terms)



Observations

- Our variant was faster for all degrees of polynomials *h* ◉ 30 than the variant of [**Savaş *et al.***]
- As *h* increases – and for sufficiently large *D*s – our variant's performance ***degrades*** due to

  (a) #iterations to find *p* 🕔 2*h* **(our variant)**

  vs.

  #iterations to find *p* 🕔$300h/\sqrt{D}$ [**Savaş *et al.***]

  (b) Root finding procedure of NTL used by [**Savaş *et al.***] is faster than ours.
- Resource requirements not too prohibitive for ***on line*** generation of Weber polynomials on hardware devices
- Combine ***on-line*** and ***off-line*** generation of polynomials

Future Work

- Adaptation of (part of) our library for various popular ***hardware devices*** (e.g. reconfigurable architectures of FPGA + processor on a chip)
- Implementation ***of the CM method*** on a variety of hardware devices and comparative study of resulting time and memory requirements
- Feasibility of a complete EC libraryon hardware devices that can ***modify*** EC system parameters