Chair for Communication Security
Ruhr University of Bochum

Hyperelliptic Curve Cryptosystems

Closing the Performance Gap to Elliptic Curves

Jan Pelzl, Thomas Wollinger, Jorge Guajardo, Christof Paar

Why use Hyperelliptic Curve Cryptosytems?

- The word „Hyperelliptic Curve Cryptosystem“ sounds awesome and impressive!
- Increasing diversity of „secure“ PK algorithms
- Shorter bitlengths have implementational advantages compared to RSA or ECC
- Perfectly suited for constraint environments

Prominent PK Schemes:

	Typical operand bitlength:
RSA	1024…2048 bit
Diffie-Hellman	1024…2048 bit
Elliptic Curves	160…256 bit

➡ Hyperelliptic curves allow for operand lengths 50…80 bit

Mathematical Preliminaries

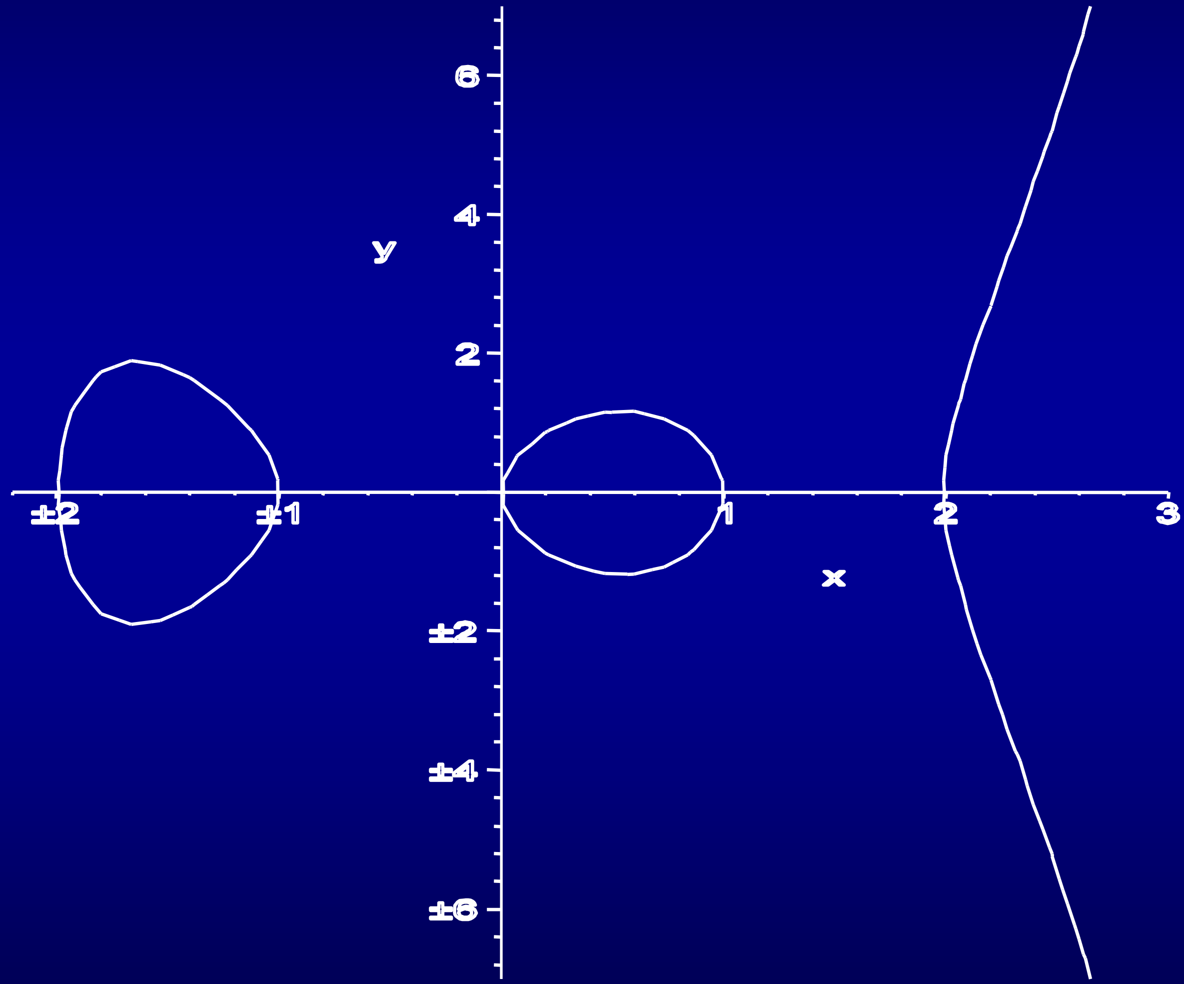
What is a hyperelliptic curve?

A HEC of genus g over a finite field $\mathbf{F}$ is given by the set of solutions $(x,y) \in \mathbf{F} \times \mathbf{F}$ to the equation

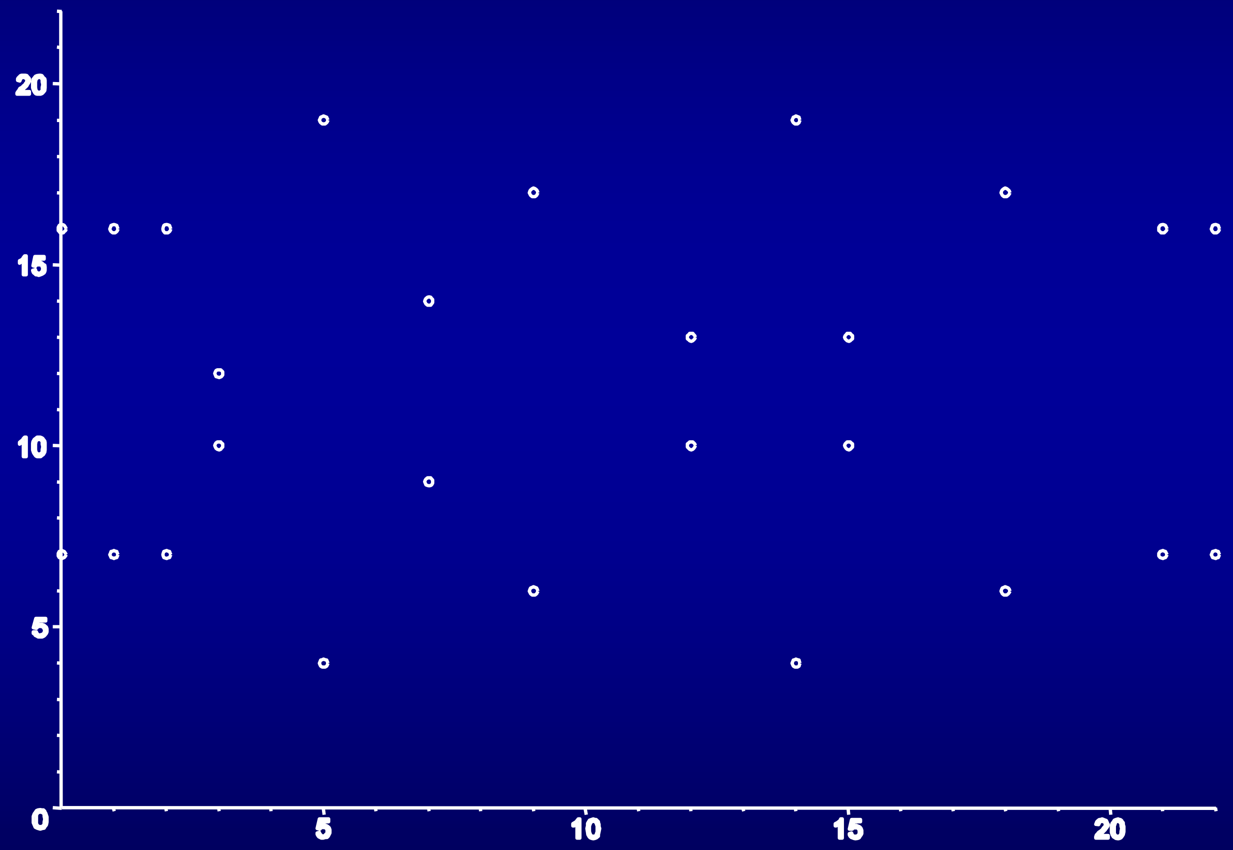
$$y^2 + h(x)y = f(x)$$

where
- $h(x)$ is a polynomial of degree $\leq g$ over $\mathbf{F}$
- $f(x)$ is a monic polynomial of degree $2g+1$ over $\mathbf{F}$
- certain further conditions

Example: C: $y^2 = x^5 - 5x^3 + 4x + 3$ over $\mathbb{R}$



Example: C: $y^2 = x^5 - 5x^3 + 4x + 3$ over $\mathbb{F}_{23}$



The group G:

Groupelement (divisor) ~ function of *g* points:

$$D = f(P_1, \ldots, P_g) = \sum_{i=1}^{g} m_{P_i} P_i$$

A divisor class group consisting of all (reduced) divisors forms the Jacobian of the curve $\mathbf{J}_C(\mathbf{F_q})$ (abelian group).

Cardinality of the group G:

- Assuming HEC of genus g over $\mathbf{F}_q$, where $q=p^n$,
- have ~q^g possible divisors since $D = f(P_1,\ldots,P_g)$

The cardinality of $\mathbf{J}_C(\mathbf{F}_\mathbf{q})$ is given by Hasse-Weil:

$$\left\lceil (\sqrt{q}-1)^{2g} \right\rceil \leq \left| J_C(F_q) \right| \leq \left\lfloor (\sqrt{q}+1)^{2g} \right\rfloor$$

E.g. want $|\mathbf{J}_C(\mathbf{F}_\mathbf{q})| \sim 2^{160}$

➔ for $g=1$ (EC) use $\mathbf{F}_{2^{160}}$

➔ for $g=2$ use $\mathbf{F}_{2^{80}}$

➔ for $g=3$ use $\mathbf{F}_{2^{53}}$

➔ for $g=4$ use $\mathbf{F}_{2^{40}}$

Do not choose genus ≥ 5 because of certain attacks and index calculus [Frey Rück, Gaudry, Thériault…]

The group law (Cantor):

- Use polynomial representation [Mumford] of divisors:

  D = div(a,b) with polynomials a(x), b(x),
  s.th. $\deg(b) \leq \deg(a) \leq g$

Cantor's Algorithm:

Input: $D_1 = \mathrm{div}(a_1,b_1)$, $D_2 = \mathrm{div}(a_2,b_2)$

Output: $D_3 = D_1 + D_2 = \mathrm{div}(a_3,b_3)$

Composition step:
$d = \gcd(a_1,a_2,b_1+b_2+h) = s_1a_1+s_2a_2+s_3(b_1+b_2+h)$
$a'_3 = a_1a_2/d$
$b'_3 = [s_1a_1b_2+s_2a_2b_1+s_3(b_1b_2+f)]/f \mod a'_3$

Reduction step:
WHILE $\deg(a'_k) > g$, DO
  $a'_k = f - b'_{k-1} \mod a'_k$
  $b'_k = (-h-b'_{k-1}) \mod a'_k$
END WHILE
$a_3 = a'_k$
$b_3 = b'_k$

Need polynomial gcd, division, multiplication and reduction!

Improvements

Observation:

Cantor‘s Algorithm slow due to polynomial arithmetic

Solution:

Transform polynomial operations into field operations (explicit formulae) by considering most frequent case (occurs with probability ~ $1\text{-}O(1/q)$) [Harley 2000]

Brief History of HECC:

1988 Use of HEC as a cryptosystem first suggested
[Koblitz 1988]

1994- Explicit formulae suggested for genus-2 HECC
[Spallek 1994; Harley 2000]

2001- Efficient explicit formulae for genus-2 HECC
[Matsuo et al. 2001; Miyamoto et al. 2002; Lange 2002]

2002- Efficient explicit formulae for genus-3 HECC
[Kuroki et al. 2002; P. 2002; this work]

2003- Efficient explicit formulae for genus-4 HECC
[P. et al. 2003]

Example: Adding divisors on HEC of genus 3

Polynomial arithmetic:

Input: $D_1 = \mathrm{div}(a_1,b_1)$, $D_2 = \mathrm{div}(a_2,b_2)$

Output: $D_3 = D_1 + D_2 = \mathrm{div}(a_3,b_3)$

Composition step: $d = \gcd(a_1,a_2,b_1+b_2+h)=s_1a_1+s_2a_2+s_3(b_1+b_2+h)$

$a'_3 = a_1a_2/d$

$b'_3 = [s_1a_1b_2+s_2a_2b_1+s_3(b_1b_2+f)]/f \mod a'_3$

Reduction step: WHILE $\deg(a'_k) > g$, DO

$a'_k = f - b'_{k-1} \mod a'_k$

$b'_k = (-h-b'_{k-1}) \mod a'_k$

END WHILE

$a_3 = a'_k$

$b_3 = b'_k$

Explicit formulae (field arithmetic only):

```
t1 = a*e;
t2 = b*d;
t3 = b*f;
t4 = c*e;
t5 = a*f;
t6 = c*d;
t7 = sqr(c+f);
t8 = sqr(b+e);
t9 = (a+d)*(t3+t4);
t10= (a+d)*(t5+t6);
r =(f+c+t1+t2)*(t7+t9) + t10*(t5+t6) + t8*(t3+t4);
t11 = (b+e)*(c+f);
inv2 = (t1+t2+c+f)*(a+d)+t8;
inv1 = inv2*d + t10 + t11;
inv0 = inv2*e + d*(t10+t11) + t9 + t7;
t12 = (inv1+inv2)*(k+n+l+o);
t13 = (l+o)*inv1;
t14 = (inv0+inv2)*(k+n+m+p);
t15 = (m+p)*inv0;
t16 = (inv0+inv1)*(l+o+m+p);
t17 = (k+n)*inv2;
rs0 = t15;
rs1 = t13+t15+t16;
rs2 = t13+t14+t15+t17;
rs3 = t12+t13+t17;
rs4 = t17;
t18 = rs3+rs4*d;
s0s = rs0 + f*t18;
s1s = rs1 + rs4*f + e*t18;
s2s = rs2 + rs4*e + d*t18;
w1 = inv(r*s2s);
w2 = r*w1;
w3 = w1*sqr(s2s);
w4 = r*w2;
w5 = sqr(w4);
```

```
s0 = w2*s0s;
s1 = w2*s1s;
s2 = w2*s2s;
z0 = s0*c;
z1 = s1*c+s0*b;
z2 = s0*a+s1*b+c;
z3 = s1*a+s0+b;
z4 = a+s1;
z5 = to_GF2E(1L);
t1 = w4*h2;
t2 = w4*h3;
u3s = d + z4 + s1;
u2s = d*u3s + e + z3 + s0 + t2 + s1*z4;
u1s = d*u2s + e*u3s + f + z2 + t1 + s1*(z3+t2) + s0*z4 + w5;
u0s = d*u1s + e*u2s + f*u3s + z1 + w4*h1 + s1*(z2+t1)
                 + s0*(z3+t2) + w5*(a+f6);
t1 = u3s+z4;
v0s = w3*(u0s*t1 + z0) + h0 + m;
v1s = w3*(u1s*t1 + u0s + z1) + h1 + l;
v2s = w3*(u2s*t1 + u1s + z2) + h2 + k;
v3s = w3*(u3s*t1 + u2s + z3) + h3;
a3 = f6 + u3s + v3s*(v3s+h3);
b3 = u2s + a3*u3s + f5 + v3s*h2 + v2s*h3;
c3 = u1s + a3*u2s + b3*u3s + f4 + v2s*(v2s+h2) + v3s*h1 + v1s*h3;
k3 = v2s + (v3s+h3)*a3 + h2;
l3 = v1s + (v3s+h3)*b3 + h1;
m3 = v0s + (v3s+h3)*c3 + h0;
```

Achieved speed-up for group operations on genus-3 curves:

	Type	# (inversion)	# (mult./squ.)	Savings[2)]
Adding	Polynomial Cantor[1)]	4	200	
	Explicit	**1**	**76**	**64%**
Doubling	Polynomial Cantor[1)]	4	207	
	Explicit	**1**	**71**	**67%**

All numbers refer to formulas for curves over odd characteristic

1) Cantor's Algorithm implemented by [Nagao 2000]

2) one inversion costs approx. 8 multiplications

In special cases 80% less computational cost!

Required field operations per group addition compared to ECC:

Genus	# (inversion)	# (mult./squ.)
1[1] (ECC)	-	16
2[2]	1	25
3[2]	1	76
4[2]	2	164

1) ECC with projective coordinates GF(p)
2) HEC over fields of arbitrary characteristic

Can HECC be faster than ECC?

Theoretical Analysis:

Given: - Microprocessor (wordsize *w*)
- Field library (ratio of multiplications per inversion = *MI*-ratio)

Goal: determine if ECC or HECC will be faster,
i.e., find accurate metric for practical purposes

Theoretical Analysis (cont.):

Methodology:

1. Express all computational expensive operations in terms of atomic operations (AOP).
2. Consider fields $\mathbf{F}_{2^n}$.
3. Use fast field multiplication algorithm [Lopez and Dahab 2000]. (Requires [$w/2+(n/4+27)\lceil n/w \rceil-7$] AOPs per field multiplication)
4. Express cost of field inversion in terms of field multiplications (MI-ratio).

Theoretical Analysis (cont.):

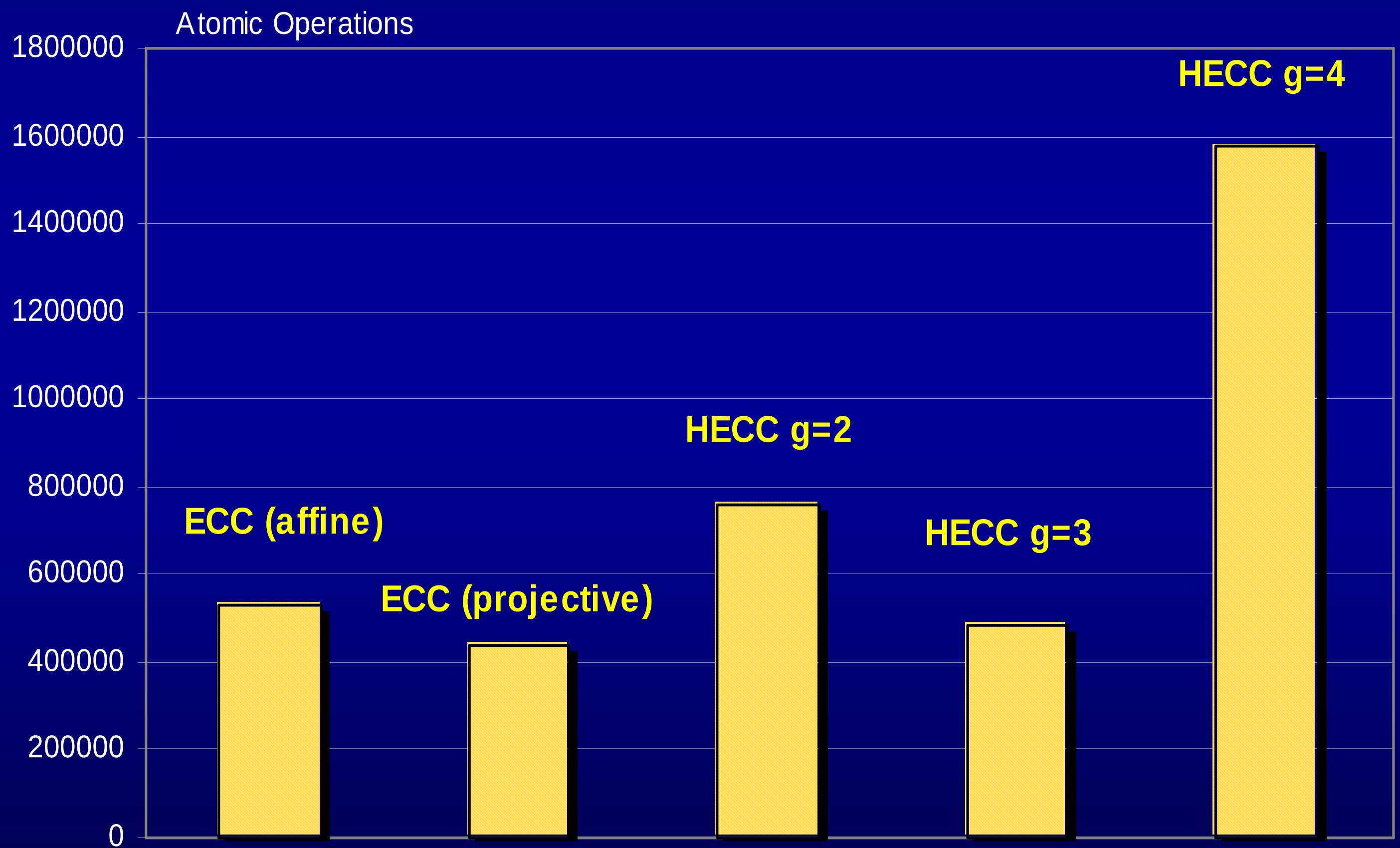
	ECC		HECC		
	affine	projective	genus-2 h(x)=x	genus-3 h(x)=1	genus-4 h(x)=x
Addition	(2+m)T	15T	(22+m)T	(65+m)T	(148+2m)T
Doubling	(2+m)T	5T	(17+m)T	(14+m)T	(75+2m)T

T := [w/2+(n/4+27)s-7]
m := MI-ratio of field library

Total numbers depend on processor type and field library!

Theoretical Analysis (result):

Number of atomic operations for 160-bit scalarmultiplication over $\mathbf{F}_{2^m}$, no special automorphisms used:



Implementation of efficient curves over fields of characteristic 2

The cost of one inversion is assumed to be approx. 6 multiplications

Implementation

Embedded performance (ARM7@80MHz):

Genus	Group order	Field	Divisor multiplication in ms
1	2^{191}	$F_{2^{191}}$	100.01
2	2^{190}	$F_{2^{95}}$	121.49
3	2^{189}	$F_{2^{63}}$	72.09
4	2^{188}	$F_{2^{47}}$	201.89

Implementation of special curves over fields of characteristic 2, no special endomorphisms used;
parts of the library by Koç et al. were used [Koç 2000]

Desktop performance (P4@1.8GHz):

Genus	Group order	Field	Divisor multiplication in ms
1	2^{191}	$\mathbf{F}_{2^{191}}$	2.78
2	2^{190}	$\mathbf{F}_{2^{95}}$	4.47
3	2^{189}	$\mathbf{F}_{2^{63}}$	3.01
4	2^{188}	$\mathbf{F}_{2^{47}}$	8.05

Implementation of special curves over fields of characteristic 2, no special endomorphisms used

Summary:

- Improved explicit formulae for genus-3 HECC
- First implementation on embedded µP
- On embedded processors, genus-3 HECC can outperform ECC and other HECC (g=2,4)
- Proposed new accurate metric for practical purposes

Further Research:

- Further optimization of genus-3 formulae (?)
- High-speed implementations for GF(p)
- Standardization of HECC/ curves
- Parallalization of HECC operations

Additional information, newest results and source code available at:

http://www.hecc.rub.de

Questions?

References

Harley, R. 2000. Fast Arithmetic on Genus Two Curves. Available at *http://cristal.inria.fr/harley/hyper/.adding.txt* and *.doubling.c*

Koblitz, N. 1988. A Family of Jacobians Suitable for Discrete Log Cryptosystems. In *Advances in Cryptology – Crypto '88*, Shafi Goldwasser, Ed. Lecture Notes in Computer Sciences, vol. 403. Springer-Verlag, Berlin, 94-99.

Koç, Ç., and Saldamli, G. 2002. Support for field arithmetic library and elliptic curve routines.

Kuroki, J., Gonda, M., Matsuo, K., Chao, J., and Tsuji, S. 2002. Fast Genus Three Hyperelliptic Curve Cryptosystems. In *The 2002 Symposium on Cryptography and Information Security, Japan – SCIS 2002*.

Lange, T. 2002. Efficient Arithmetic on Genus Two Hyperelliptic Curves over Finite Fields via Explicit Formulae. Cryptology ePrint Archive, Report 2002/121. *http://eprint.iacr.org/*

Lopez, J., and Dahab, R. 2000. High-speed software multiplication in F_2m. In *INDOCRYPT 2000, 203-212*.

Matsuo, K., Chao, J., and Tsuji, S. 2001. Fast Genus Two Hyperelliptic Cryptosystems. In *ISEC2001-31, IEICE*.

Miyamoto, Y., Doi, H., Matsuo, K., Chao, J., and Tsuji, S. 2002. A Fast Addition Algorithm of Genus Two Hyperelliptic Curve. In *The 2002 Symposium on Cryptography and Information Security – SCIS 2002, IEICE Japan*. 497-502, in Japanese.

Nagao, K. 2002. Improving Group Law Algorithms for Jacobians of Hyperelliptic Curves. In *ANTS IV*, W. Bosma, Ed. Lecture Notes in Computer Science, vol. 1838. Springer-Verlag, Berlin, 439-448.

Pelzl, J. 2002. Hyperelliptic Cryptosystems on Embedded Processors. M.S. thesis, Department of Electrical Engineering and Information Sciences, Ruhr-Universität Bochum, Bochum, Germany.

Pelzl, J., Wollinger, T., Guajardo, J., and Paar, C. 2003. Hyperelliptic Curve Cryptosystems: Closing the Performance Gap to Elliptic Curves. In *Workshop on Cryptographic Harware and Embedded Systems 2003 - CHES 2003*.

Pelzl, J., Wollinger, and Paar, C. 2003. Low Cost Security: Explicit Formulae for Genus-4 Hyperelliptic Curves. In *Selected Areas in Cryptography 2003 - SAC 2003*.

Spallek, A.M. 1994. Kurven vom Geschlecht 2 und ihre Anwendung in Public-Key-Kryptosystemen, 1994. PhD Thesis. Universität Gesamthochschule Essen.

Thériault, N. 2003. Index calculus attack for hyperelliptic curves of small genus, preprint 2003. University of Toronto.