Published in *Proc. of the 3rd Workshop on Mathematical Cryptology (WMC 2012) and 3rd International Conference on Symbolic Computation and Cryptography (SCC 2012)*, pp. 45-50, Castro Urdiales, Spain, July 2012.

UNIFIED ADDITION FORMULÆ FOR HYPERELLIPTIC CURVE CRYPTOSYSTEMS

OUMAR DIAO AND MARC JOYE

1. Introduction

Hyperelliptic curve cryptography was introduced by Koblitz in 1989 [8] (see also [9]) as an alternative to elliptic curve cryptography. It bases its security on the discrete logarithm problem in the Jacobian of an hyperelliptic curve of genus $g \geq 2$ (HCDLP). Recent cryptanalytic results [7] have shown that hyperelliptic curve cryptosystems of genus $g \geq 3$ are prone to attacks better than generic methods for solving the HCDLP. As a consequence, although our techniques readily apply to any genus, the focus will be put on genus-2 hyperelliptic curves.

In practice, the hardness of the HCDLP is not sufficient (but necessary) to guarantee the security of the underlying cryptosystems; it only provides black-box security. An attacker may have more information than a mere access to the input and output of the algorithms. Specifically, the attacker may monitor the execution of the algorithm and get additional information through certain side channels such as the running time [10] or the power consumption [11]. Of particular importance is the resistance against simple side-channel analysis. Resistance against the more sophisticated differential side-channel analysis can be achieved using various randomization techniques [1]. This paper presents unified addition formulæ for hyperelliptic curve cryptosystems as an efficient means to thwart simple side-channel attacks, extending the techniques of [3, 2] to genus $g > 1$.

2. Background on Hyperelliptic Curves

A *hyperelliptic curve of genus g over a field* $\mathbb{K}$ is a non-singular curve given by an equation

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

where $f \in \mathbb{K}[x]$ is a monic polynomial of degree $2g + 1$ and $h \in \mathbb{K}[x]$ is a polynomial of degree $\leq g$. The set of $\mathbb{K}$-rational points on C, denoted $C(\mathbb{K})$, is the set of all points $(x, y) \in \mathbb{K} \times \mathbb{K}$ satisfying the above equation together with the so-called 'point at infinity' ∞. The opposite of a finite point $P = (a, b)$ is the point $-P = (a, -b - h(a))$ and $-\infty = \infty$.

A *divisor on C* is a finite formal sum $D = \sum_{P \in C(\overline{\mathbb{K}})} n_P(P)$ with $n_P \in \mathbb{Z}$; its degree is defined as $\sum n_P$. A divisor D is said defined over $\mathbb{K}$ if $D = \sum n_P(P^\sigma)$ for every automorphism σ of $\overline{\mathbb{K}}$ over $\mathbb{K}$. The *function field of C over* $\mathbb{K}$, denoted $\mathbb{K}(C)$, is the field of fractions of the polynomial ring $\mathbb{K}[C] = \mathbb{K}[x, y]/(y^2+h(x)y-f(x))$. Similarly, the function field $\overline{\mathbb{K}}(C)$ is defined as the field of fractions of $\overline{\mathbb{K}}[C]$. To any nonzero rational function $\psi \in \overline{\mathbb{K}}(C)$, one can associate a divisor via the valuation at all points as $\mathrm{div}(\psi) = \sum_{P \in C(\overline{K})} \nu_P(\psi)(P)$. Such a divisor is called a *principal divisor* and is of degree 0. The set of divisors defined over $\mathbb{K}$ forms an additive group denoted Div_C. The subgroup of degree-0 divisors is denoted Div^0_C and its subgroup of principal divisors is denoted Princ_C. The *Jacobian of the curve C* is the quotient group $J_C = \mathrm{Div}^0_C \,/\, \mathrm{Princ}_C$. Riemann-Roch theorem tells us that each element of J_C can be uniquely represented by a *reduced divisor*, that is, a divisor of the form

$$D = \sum_{i=1}^{m} (P_i) - m(\infty)$$

with (i) $P_i \neq \infty$, (ii) $P_i \neq -P_j$ if $i \neq j$, and (iii) $m \leq g$. A divisor satisfying Conditions (i) and (ii) (but not necessarily Condition (iii)) is said *semi-reduced*.

To avoid working in an extension of $\mathbb{K}$, a semi-reduced divisor $D = \sum_{i=1}^{m} (P_i) - m(\infty)$ is preferably identified using *Mumford representation* as a pair of polynomials $u(x)$ and $v(x)$ in $\mathbb{K}[x]$ where, letting $P_i = (x_i, y_i)$,

- $u := u(x) = \prod_{i=1}^{m} (x - x_i)$, and

- $v := v(x)$ is the unique polynomial of degree $< m$ such that $v(x_i) = y_i$ with appropriate multiplicity when P_i appears more than once in D.

We write $D = [u, v]$. Mumford representation leads to efficient algorithms for adding or doubling group elements in J_C [4].

Explicit formulæ for genus-2 hyperelliptic curves are detailed in [12]. The formulæ were subsequently improved by Costello and Lauter through a more direct geometric interpretation of the group law. Letting M, S and I the respective costs of a multiplication, squaring and inversion in $\mathbb{K}$, the best operation counts are $\underline{1\mathsf{I} + 17\mathsf{M} + 4\mathsf{S}}$ for the addition in J_C and $\underline{1\mathsf{I} + 19\mathsf{M} + 6\mathsf{S}}$ for the doubling in J_C [5].

3. Unified Addition Formulæ

Classically, computing in Jacobians of hyperelliptic curves is carried out as an application of Cantor's algorithm [4]. It takes on input two reduced divisors in Mumford representation and outputs a reduced divisor in Mumford representation. In more detail, given two reduced divisors $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$, the algorithm first produces a semi-reduced divisor $[u, v]$ equivalent to $D_1 + D_2$ modulo Princ_C, such that

$$u = \frac{u_1 u_2}{d^2} \quad \text{and} \quad v \equiv \frac{s_1 u_1 v_2 + s_2 u_2 v_1 + s_3(v_1 v_2 + f)}{d} \pmod{u} \tag{1}$$

with $d = \gcd(u_1, u_2, v_1 + v_2 + h) = s_1 u_1 + s_2 u_2 + s_3(v_1 + v_2 + h)$ for polynomials $s_1, s_2, s_3 \in \mathbb{K}[x]$ given by the extended Euclidean algorithm. This divisor is then reduced in a second step by repeatedly applying

$$u \leftarrow \mathrm{Monic}\left(\frac{v^2 + vh - f}{u}\right) \quad \text{and} \quad v \leftarrow -v - h \pmod{u} \ .$$

until $\deg(u) \leq g$.

For computational purposes, there are two main cases to consider:

(1) Cantor general doubling: $D_1 = D_2$ and $\gcd(u_1, 2v_1 + h) = 1$;
(2) Cantor general addition: $D_1 \neq D_2$ and $\gcd(u_1, u_2) = 1$.

Distinguishing these two cases allows one to derive explicit formulæ for low-genus curves. As shown in [4], the expression for v then verifies the simpler equation

$$v \equiv v_1 + s_3(f - v_1 h - {v_1}^2) \pmod{u} \quad \text{and} \quad v \equiv v_1 + s_1 u_1 (v_2 - v_1) \pmod{u} \tag{2}$$

for a Cantor general doubling and a Cantor general addition, respectively. The next proposition is our main ingredient. It states a relation that is satisfied for both cases. This will be useful in designing unified addition formulæ.

Proposition 1. *Using the previous notation, let $D_1 = [u_1, v_1]$ and $D_2 = [u_2, v_2]$ be two reduced divisors in a general Cantor operation. Then $[u, v] \sim D_1 + D_2$ where*

$$u = u_1 u_2 \quad \textit{and} \quad v(h + v_1 + v_2) \equiv f + v_1 v_2 \pmod{u} \ . \tag{3}$$

Proof. See e.g. [6, Proof of Theorem 10.3.14]. □

We now develop explicit addition formulæ in the Jacobian of genus-2 curves. We are concerned with the frequent case involving divisors of full degree. So, for $i \in \{1, 2\}$, we let $u_i := u_i(x) = x^2 + u_{i,1}x + u_{i,0}$ and $v_i := v_i(x) = x^2 + v_{i,1}x + v_{i,0}$. We also let $v := v(x) = \ell_3 x^3 + \ell_2 x^2 + \ell_1 x + \ell_0$ for unknown coefficients ℓ_j, $0 \leq j \leq 3$. As in [5], we build a system of linear equations that solves to give these coefficients ℓ_j.

From Eq. (2), it clearly appears that in both cases (i.e., doubling and addition) $v \equiv v_1 \pmod{u_1}$ — remember that $u_1 \mid (f - v_1 h - {v_1}^2)$. This can be rewritten as

$$\ell_3 x^3 + \ell_2 x^2 + \ell_1 x + \ell_0 - (v_{1,1}x + v_{1,0}) \equiv 0 \pmod{(x^2 + u_{1,1}x + u_{1,0})} \, ,$$

which gives rise to two linear equations:

$$\begin{cases} (u_{1,1}^2 - u_{1,0})\ell_3 - u_{1,1}\ell_2 + \ell_1 = v_{1,1} \\ u_{1,1}u_{1,0}\ell_3 - u_{1,0}\ell_2 + \ell_0 = v_{1,0} \end{cases} ,$$

or equivalently,

$$\begin{pmatrix} u_{1,1}^2 - u_{1,0} & -u_{1,1} & 1 & 0 \\ u_{1,1}u_{1,0} & -u_{1,0} & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \ell_3 \\ \ell_2 \\ \ell_1 \\ \ell_0 \end{pmatrix} = \begin{pmatrix} v_{1,1} \\ v_{1,0} \end{pmatrix} \quad . \tag{4}$$

Further linear equations are obtained from the second relation in Eq. (3). We have $u_1u_2 := u_1(x)u_2(x) = x^4 + (u_{1,1} + u_{2,1})x^3 + (u_{1,0} + u_{2,0} + u_{1,1}u_{2,1})x^2 + (u_{1,1}u_{2,0} + u_{1,0}u_{2,1})x + u_{1,0}u_{2,0}$. Hence, letting $f := f(x) = x^5 + f_4x^4 + f_3x^3 + f_2x^2 + f_1x + f_0$ and $h := h(x) = h_2x^2 + h_1x + h_0$, we get after a little algebra $f + v_1v_2 \bmod u_1u_2 := F_3x^3 + F_2x^2 + F_1x + F_0$ with

$$\begin{aligned}
F_3 &= u_{1,1}^2 + u_{2,1}^2 + u_{1,1}u_{2,1} - (u_{1,0} + u_{2,0}) - f_4(u_{1,1} + u_{2,1}) + f_3 \\
F_2 &= (u_{1,1} + u_{2,1} - f_4)(u_{1,0} + u_{,20} + u_{1,1}u_{2,1}) - (u_{1,1}u_{2,0} + u_{1,0}u_{2,1}) + f_2 + v_{1,1}v_{2,1} \\
F_1 &= (u_{1,1} + u_{2,1} - f_4)(u_{1,1}u_{2,0} + u_{1,0}u_{2,1}) - u_{1,0}u_{2,0} + f_1 + v_{1,1}v_{2,0} + v_{1,0}v_{2,1} \\
F_0 &= (u_{1,1} + u_{2,1} - f_4)u_{1,0}u_{2,0} + f_0 + v_{1,0}v_{2,0}
\end{aligned}$$

and $v(h + v_1 + v_2) \bmod u_1u_2 := L_3x^3 + L_2x^2 + L_1x + L_0$ with

$$\begin{aligned}
L_3 &= [h_2(u_{1,1}^2 + u_{2,1}^2 + u_{1,1}u_{2,1} - u_{1,0} - u_{2,0}) + H_0 - (u_{1,1} + u_{2,1})H_1]\ell_3 + \\
&\qquad [H_1 - h_2(u_{1,1} + u_{2,1})]\ell_2 + h_2\ell_1 \\
L_2 &= [h_2((u_{1,1} + u_{2,1})(u_{1,0} + u_{2,0} + u_{1,1}u_{2,1}) - (u_{1,1}u_{2,0} + u_{1,0}u_{2,1})) - \\
&\qquad H_1(u_{1,0} + u_{2,0} + u_{1,1}u_{2,1})]\ell_3 + [H_0 - h_2(u_{1,0} + u_{2,0} + u_{1,1}u_{2,1})]\ell_2 + H_1\ell_1 + h_2\ell_0 \\
L_1 &= [h_2((u_{1,1} + u_{2,1})(u_{1,1}u_{2,0} + u_{1,0}u_{2,1}) - u_{1,0}u_{2,0}) - H_1(u_{1,1}u_{2,0} + u_{1,0}u_{2,1})]\ell_3 + \\
&\qquad h_2(u_{1,1}u_{2,0} + u_{1,0}u_{2,1})\ell_2 + H_0\ell_1 + H_1\ell_0 \\
L_0 &= [h_2(u_{1,1} + u_{2,1})u_{1,0}u_{2,0} - H_1u_{1,0}u_{2,0}]\ell_3 - h_2u_{1,0}u_{2,0}\ell_2 + H_0\ell_0
\end{aligned}$$

where $H_1 = h_1 + v_{1,1} + v_{2,1}$ and $H_0 = h_0 + v_{1,0} + v_{2,0}$.

The previous relations hold over a field of any characteristic. In order to get a fair comparison with the best operation count in [5], we henceforth suppose that the underlying field $\mathbb{K}$ is such that $\mathrm{Char}\,\mathbb{K} \neq 2, 5$, in which case we can assume without loss of generality $h_2 = h_1 = h_0 = 0$ and $f_4 = 0$. The expressions for F_j and L_j then have a simpler form and, combining with Eq. (4), the previous relations become

$$\begin{pmatrix} u_{1,1}^2 - u_{1,0} & -u_{1,1} & 1 & 0 \\ u_{1,1}u_{1,0} & -u_{1,0} & 0 & 1 \\ H_0 - (u_{1,1} + u_{2,1})H_1 & H_1 & 0 & 0 \\ -H_1(u_{1,0} + u_{2,0} + u_{1,1}u_{2,1}) & H_0 & H_1 & 0 \\ -H_1(u_{1,1}u_{2,0} + u_{1,0}u_{2,1}) & 0 & H_0 & H_1 \\ -H_1u_{1,0}u_{2,0} & 0 & 0 & H_0 \end{pmatrix} \cdot \begin{pmatrix} \ell_3 \\ \ell_2 \\ \ell_1 \\ \ell_0 \end{pmatrix} = \begin{pmatrix} v_{1,1} \\ v_{1,0} \\ F_3 \\ F_2 \\ F_1 \\ F_0 \end{pmatrix} \quad . \tag{5}$$

Multiplying row 1 by $-H_1$ and adding the resulting row to row 4 yields the smaller system

$$\begin{pmatrix} H_0 - (u_{1,1} + u_{2,1})H_1 & H_1 \\ -H_1(u_{1,1}^2 + u_{2,0} + u_{1,1}u_{2,1}) & H_0 + H_1u_{1,1} \end{pmatrix} \cdot \begin{pmatrix} \ell_3 \\ \ell_2 \end{pmatrix} = \begin{pmatrix} F_3 \\ F_2 - H_1v_{1,1} \end{pmatrix} \tag{6}$$

that can be solved for ℓ_3 and ℓ_2. The values of ℓ_1 and ℓ_0 can then be obtained from Eq. (4). The next step consists in reducing the so-obtained divisor $[u, v]$ to get $[\widetilde{u}, \widetilde{v}] = [u_1, v_1] + [u_2, v_2]$. Letting $\widetilde{u} := \widetilde{u}(x) = x^2 + \widetilde{u}_{11}x + \widetilde{u}_{10}$ and $\widetilde{v} := \widetilde{v}(x) = \widetilde{v}_{11}x + \widetilde{v}_{10}$, this can be achieved as presented in [5]; i.e.,

$$\begin{aligned}
\widetilde{u}_{11} &= -(u_{1,1} + u_{2,1}) - (1 - 2\ell_2\ell_3)/\ell_3^2\,, \\
\widetilde{u}_{10} &= -\big(u_{1,0} + u_{2,0} + u_{1,1}u_{2,1} + (u_{1,1} + u_{2,1})\widetilde{u}_{1,1}\big) + (2\ell_1\ell_3 + \ell_2^2)/\ell_3^2\,, \\
\widetilde{v}_{11} &= -\big(\ell_3(\widetilde{u}_{1,1}^2 - \widetilde{u}_{1,0}) - \ell_2\widetilde{u}_{1,1} + \ell_1\big)\,, \\
\widetilde{v}_{10} &= -(\ell_3\widetilde{u}_{1,1}\widetilde{u}_{1,0} - \ell_2\widetilde{u}_{1,0} + \ell_0)\ .
\end{aligned}$$

Altogether our unified addition algorithm can be evaluated using only $\underline{1\mathsf{I} + 21\mathsf{M} + 6\mathsf{S}}$. A detailed Magma implementation is provided in Appendix A.

4. Conclusion

This paper presented efficient unified addition formulæ for hyperelliptic curve cryptography. Interestingly, the proposed formulæ only slightly increase the complexity and therefore provide a cost-efficient way to prevent simple side-channel attacks.

References

[1] R. M. Avanzi. Countermeasures against differential power analysis for hyperelliptic curve cryptosystems. In C. D. Walter et al., editors, *Cryptographic Hardware and Embedded Systems – CHES 2003*, volume 2779 of *Lect. Notes in Comp. Sci.*, pages 366–381. Springer, 2003.

[2] É. Brier, I. Déchène, and M. Joye. Unified point addition formulæ for elliptic curve cryptosystems. In N. Nedjah and L. de Macedo, editors, *Embedded Cryptographic Hardware: Methodologies & Architectures*, pages 247–256. Nova Science Publishers, 2004.

[3] É. Brier and M. Joye. Weierstraß elliptic curves and side-channel attacks. In D. Naccache and P. Paillier, editors, *Public Key Cryptography*, volume 2274 of *Lect. Notes in Comp. Sci.*, pages 335–345. Springer-Verlag, 2002.

[4] D. G. Cantor. Computing in the Jacobian of a hyperelliptic curve. *Math. Comp.*, 48(177):95–101, 1987.

[5] C. Costello and K. Lauter. Group law computations on Jacobians of hyperelliptic curves. Cryptology ePrint Archive, Report 2011/306, 2011. `http://eprint.iacr.org/`.

[6] S. D. Galbraith. *Mathematics of Public Key Cryptography*. Cambridge University Press, 2012.

[7] P. Gaudry, E. Thomé, N. Thériault, and C. Diem. A double large prime variation for small genus hyperelliptic index calculus. *Math. Comp.*, 76(257):475–492, 2007.

[8] N. Koblitz. Hyperelliptic cryptosystems. *J. Cryptology*, 1(3):139–150, 1989.

[9] N. Koblitz. *Algebraic Aspects of Cryptography*. Springer, 1998.

[10] P. C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In N. Koblitz, editor, *Advances in Cryptology – CRYPTO '96*, volume 1109 of *Lect. Notes in Comp. Sci.*, pages 104–113. Springer-Verlag, 1996.

[11] P. C. Kocher, J. Jaffe, and B. Jun. Differential power analysis. In M. Wiener, editor, *Advances in Cryptology – CRYPTO '99*, volume 1666 of *Lect. Notes in Comp. Sci.*, pages 388–397. Springer-Verlag, 1999.

[12] T. Lange. Formulae for arithmetic on genus 2 hyperelliptic curves. *Appl. Algebra Eng. Commun. Comput.*, 15(5):295–328, 2005.

Appendix A. Magma Implementation

```
UniAddLaw := function(D1,D2)
   J := Parent(D1);
   C := Curve(J);
   Fq := BaseRing(C);
  _<x> := PolynomialRing(Fq);
   f,h := HyperellipticPolynomials(C);

   u1 := D1[1]; v1 := D1[2];
   u2 := D2[1]; v2 := D2[2];

   f2 := Coefficient(f,2); f3 := Coefficient(f,3);

   u11 := Coefficient(u1,1); u10 := Coefficient(u1,0);
   u21 := Coefficient(u2,1); u20 := Coefficient(u2,0);
   v11 := Coefficient(v1,1); v10 := Coefficient(v1,0);
   v21 := Coefficient(v2,1); v20 := Coefficient(v2,0);

   U11 := u11^2; U10 := u11*u10;
   U21 := u21^2; U20 := u21*u20;
   Su1 := u11 + u21; Su0 := u10 + u20;
```

```
   Pu1 := (Su1^2 - U11 - U21)/2; // instead of Pu1 := u11*u21;
   H1 := v11 + v21; H0 := v10 + v20;
   M1 := H0 - H1*Su1; M2 := H1;
   M3 := -H1*(U11 + Pu1 + u20); M4 := H0 + u11*H1;
   z1 := f2 + Su1*Pu1 + U10 + U20 - v11^2;
   z2 := f3 + U11 + U21 + Pu1 - Su0;

   t1 := (z1 + M3)*(z2 - M1); t2 := (z1 - M3)*(z2 + M1);
   t3 := (z1 + M4)*(z2 - M2); t4 := (z1 - M4)*(z2 + M2);
   d := t3 + t4 - t1 - t2 - 2*(M3 - M4)*(M1 + M2);

   l2 := t2 - t1; l3 := t3 - t4;

   A := 1/(d*l3); B := d*A; C := d*B; D := l2*B; E := l3^2*A; C2 := C^2;

   utilde11 := 2*D - C2 - Su1;
   utilde10 := D^2 + C*(v11 + v21) - ((utilde11 - C2)*Su1 + (U11 + U21))/2;
   Utilde11 := utilde11^2;
   Utilde10 := utilde11*utilde10;
   vtilde11 := D*(u11 - utilde11) + Utilde11 - utilde10 - U11 + u10;
   vtilde10 := D*(u10 - utilde10) + Utilde10 - U10;
   vtilde11 := -(E*vtilde11 + v11);
   vtilde10 := -(E*vtilde10 + v10);

   utilde := x^2 + utilde11*x + utilde10;
   vtilde := vtilde11*x + vtilde10;

   return J![utilde,vtilde];
end function;
```


Université de Rennes I
E-mail address: oumar.diao@univ-rennes1.fr

Technicolor
E-mail address: marc.joye@technicolor.com