Faster arithmetic on elliptic curves using F_{p^2}. Application to GLV-GLS and NIST elliptic curves over F_p isomorphic to twisted Hessian curves over fields extension.

Michał WROŃSKI
Institute of Mathematics and Cryptology
Faculty of Cybernetics
Military University of Technology in Warsaw
mwronski@wat.edu.pl

Abstract:

In this article we present how we can use fast F_{p^2} multiplication to speed-up arithmetic on elliptic curves. We use parallel computations for multiplication in F_{p^2} which is not much slower than multiplication in F_p. We show two applications of this method.

In the first we show that using twisted Edwards curves over F_{p^2} with fast computable endomorphism (GLV-GLS method) may be nowadays on of the fastest (or even the fastest) solution in hardware applications.

In the second we show how we can speed-up point scalar multiplication on NIST P-224 and NIST P-256 curves. We use field extension (F_{p^2}) to find isomorphic to these curves twisted Hessian curves over F_{p^2}. Our solution is faster than classic solutions up to 28.5% for NIST P-256 and up to 27.2% for NIST P-224 if we consider solution invulnerable for side channel attacks. We can also use different formula for point doubling and points addition and then our solution is faster up to 21.4% for NIST P-256 and up to 19.9% for NIST P-224 comparing to classic solutions.

Keywords: Field extension, twisted Hessian curves, GLV-GLS, twisted Edwards curves, point scalar multiplication

1 Introduction

In 1999 NIST published in [1] its standard elliptic curve. From 1999 till now has been published many efficient methods for point scalar multiplication on elliptic curves. Most of them are not applicable for NIST elliptic curves over prime fields.

In 2001 Gallant, Lambert and Vanstone proposed in [2] efficient method (GLV method) for fast point scalar multiplication for elliptic curves on which exist fast computable endomorphism. In 2007 Edwards published first article about his new idea of elliptic curve (Edwards curve, see [3]) and in the same year Birkner et al. gave in [4] its generalization - twisted Edwards curve. In

2009 Galbraith, Lin and Scott showed in [5] that GLV method may be applied to many elliptic curves over F_{p^2} (GLS method). In 2011 there was presented four dimensional GLV method (see [6], [7]). Today there are many other publications which extend GLV-GLS methods.

What is the most important, GLV and GLS methods allow us to speed-up point scalar multiplication only for special types of elliptic curves. In 2010 Ghosh et al. presented in [8] method of hardware parallel multiplication in F_{p^2} (in application to pairing) which is up to three times faster than classic methods of multiplication in F_{p^2}. In 2011 Cheung et al. presented in [9] fast hardware arithmetic in F_{p^2} using RNS method, also in application to pairing.

In 2010 Farashahi and Joye presented in [18] efficient arithmetic on generalized Hessian curves. In 2015 Bernstein et. all presented in [16] new method for fast arithmetic on twisted Hessian curves (twisted Hessian curves are isomorphic to generalized Hessian curves). We found that we can speed-up point scalar multiplication on NIST P-224 and NIST P-256 curves using twisted Hessian curves arithmetic over F_{p^2}. We can also use complete formula to prevent our solution against side channel attacks.

This idea bases on similar method used by us in [17], when we used arithmetic on (twisted) Edwards curve to speed-up arithmetic on NIST curves over F_p. New solution which base on twisted Hessian curves, although is not applicable to all NIST curves over F_p, ensures similar speed-up as using (twisted) Edwards curve arithmetic over F_{p^3} but requires much less resources.

In this article we present how to connect these all techniques and use them for construction of fast hardware point scalar multiplication for special types of elliptic curves.

Firstly we present how we can use twisted Edwards curves over F_{p^2} with fast computable endomorphism. Presented solution is faster than classic solutions with similar level of security from five to nine times. We present method for generation of such curves. We also propose to use this solution (or some similar) in hardware applications of elliptic curve arithmetic.

Secondly we present how we can use twisted Hessian curves arithmetic over F_{p^2} to speed-up computations on NIST P-224 and NIST P-256 curves. This method is faster than classic solutions up to 28.5% for NIST P-256 and up to 27.2% for NIST P-224.

2 Arithmetic in F_{p^2}

Every element in $A \in F_{p^2}$ may be written as $A = a_1x + a_0$, where $a_0, a_1 \in F_p$.

Let's $A, B \in F_{p^2}$. And $A = a_1x + a_0$ and $B = b_1x + b_0$. Then $A \pm B = a_1x + a_0 + b_1x + b_0 = (a_1 \pm b_1)(\mathrm{mod}\, p)x + (a_0 \pm b_0)(\mathrm{mod}\, p)$. Addition and subtraction are not costly operations. In hardware we are able to compute them in only one processor cycle. Although fast F_{p^2} arithmetic is presented in [8] to speed-up pairing, we show its different application. We also use some ideas to speed-up this arithmetic for given irreducible polynomial. We also show how we can make fast inversion of element in F_{p^2}, basing on idea presented in [11].

2.1 Choosing of irreducible polynomial

The cost of multiplication and inversion depends on form of irreducible polynomial we operate on. We prefer irreducible polynomials of form $f(t) = t^2 \pm c$, where $c \in F_p$ is small. Using such kind of irreducible polynomial decreases amount of necessary computations.

The form of irreducible polynomial depends on the kind of prime number p. Because we are interested in large primes, we need to consider three cases:

1. $p \equiv 3 (\mathrm{mod}\, 4)$. In this case it is easy to show that $f(t) = t^2 + 1$ is irreducible polynomial. If we use Legendre symbol $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} (\mathrm{mod}\, p)$ it is quite obvious that for $a = -1$ we have $\left(\frac{-1}{p}\right) \equiv -1 (\mathrm{mod}\, p)$ and it means that -1 is non-quadratic residue $\mathrm{mod}\, p$. It means that $f(t) = t^2 + 1$ is irreducible over F_p.

2. $p \equiv 5 (\mathrm{mod}\, 8)$. In this case we can show that $f(t) = t^2 - 2$ is irreducible polynomial. Using Legendre symbol we get $\left(\frac{2}{p}\right) \equiv -1 (\mathrm{mod}\, p)$ and thus $f(t) = t^2 - 2$ is irreducible over F_p.

3. $p \equiv 1 (\mathrm{mod}\, 8)$. In this case we need to find values $\pm c$ with the smallest bitlength using brutal searching method.

2.2 Multiplication

Multiplication is crucial operation in elliptic curve arithmetic. However it is not the longest operation (the longest is inversion) during scalar point multiplication we need to compute multiplication in F_{p^2} many times. Inversion we can compute only once, at the end of all computations.

Let $A, B \in F_{p^2}$ where $A = a_1 t + a_0$ and $B = b_1 t + b_0$. Then $A \cdot B = (a_1 t + a_0)(b_1 t + b_0) = a_1 b_1 t^2 + (a_1 b_0 + a_0 b_1) t + a_0 b_0$. We should remember that $t^2 = \mp c$ (because $t^2 \pm c = 0$). Then:

$C = A \cdot B = (a_1 b_0 + a_0 b_1) t + a_0 b_0 + c a_1 b_1 = \mp c_1 t + c_0$.

So:

$c_1 = a_1 b_0 + a_0 b_1$

$c_0 = a_0 b_0 \mp c a_1 b_1$

It is easy to see that we need to make 4 multiplications, 1 multiplication by small constant and 2 additions in F_p. The number of processor cycles for multiplication in F_{p^2} for $|c| > 1$is $T_M + \lceil \log_2 c \rceil + 1$.

In the case when $p \equiv 3 (\mathrm{mod}\, 4)$ we can choose $f(t) = t^2 + 1$ and then the cost of multiplication is $T_M + 1$ processor cycles. In the case $p \equiv 5 (\mathrm{mod}\, 8)$ we can choose $f(t) = t^2 - 2$ and then the cost of multiplication is $T_M + \lceil \log_2 c \rceil + 1 = T_M + 2$ processor cycles.

Fortunately using Karatsuba algorithm we are able to decrease number of multiplications.

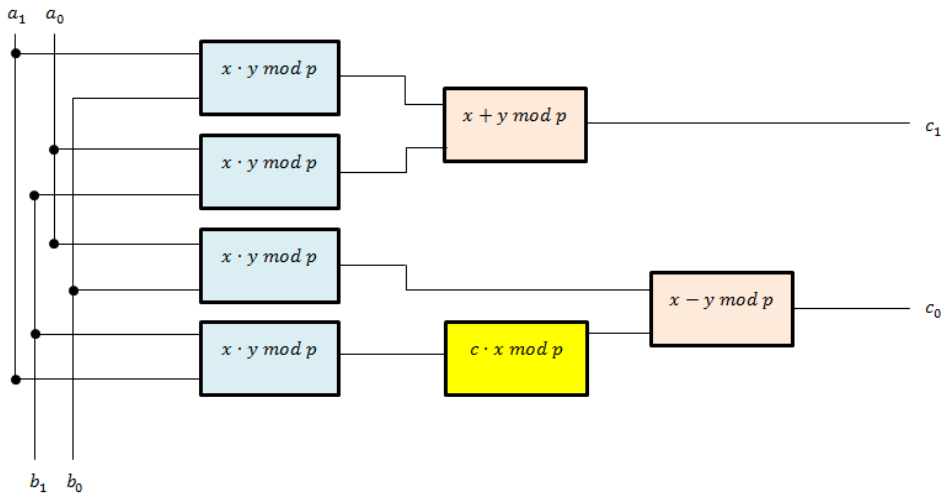

Figure 1: Scheme of multiplication method in F_{p^2} using classic method (for each block x is the upper line, y is the lower line on input)

Let $f(t) = t^2 \pm c$. Then:
$L = (a_1 + a_0)(b_1 + b_0)$
$M = a_1 b_1$
$N = a_0 b_0$
$R = L - M - N = a_1 b_0 + a_0 b_1$
Then:
$A \cdot B = (a_1 b_0 + a_0 b_1)t + a_0 b_0 \mp c a_1 b_1 = Rt \mp Mc + N$
And it easy to see that:
$c_1 = R$ and $c_2 = \mp Mc + N$.

Of course we need to count 3 multiplications, 5 additions, 1 multiplication by small constant.

Although in software applications multiplication in F_{p^2} is still much longer than multiplication in F_p, in hardware we are able to do it in almost the same time using parallelism.

The total number of processor cycles to make multiplication is $MAX\{T_M + 2, T_M + \lceil \log_2 c \rceil + 1\}$ (without initialization cycles). It is easy to see that the smaller c we choose, the less operations we need to do.

In the case when $p \equiv 3 (\mathrm{mod}\, 4)$ we can choose $f(t) = t^2 + 1$ and then the cost of multiplication is $MAX\{T_M + 2, T_M + \lfloor \log_2 1 \rfloor + 1\} = T_M + 2$ processor cycles. In the case $p \equiv 5 (\mathrm{mod}\, 8)$ we can choose $f(t) = t^2 - 2$ and then the cost of multiplication is $MAX\{T_M + 2, T_M + \lceil \log_2 2 \rceil + 1\} = T_M + 2$ processor cycles.

As we can see, although using Karatsuba method decreases the resources we need to use, in some situations increases time needed for making computations.

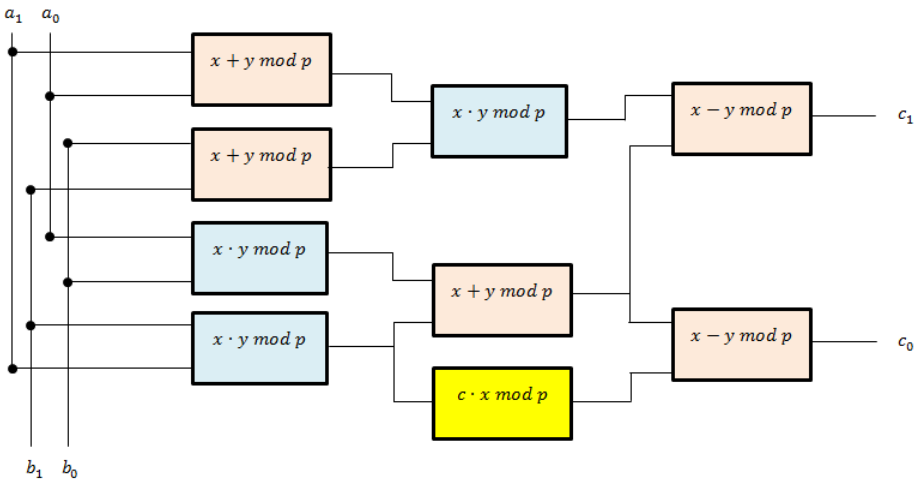

Figure 2: Scheme of multiplication in F_{p^2} using Karatsuba method for $|c| > 2$

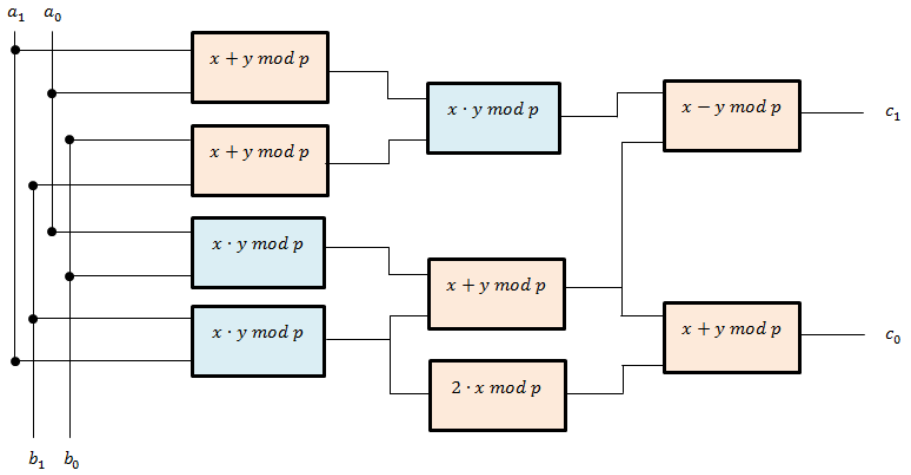

Figure 3: Scheme of multiplication in F_{p^2} using Karatsuba method for $c = -2$

2.3 Inversion

It is possible for $A \in F_{p^2}$ to count its inversion A^{-1} by count one inversion of element from F_p. We will show the method for irreducible polynomial of form $f(t) = t^2 \pm c$. The idea we base on may be found in [11].

Let's write:
$A = \begin{bmatrix} a_1 \\ a_0 \end{bmatrix}$, and $A^{-1} = \begin{bmatrix} b_1 \\ b_0 \end{bmatrix}$.

If $M = \begin{bmatrix} a_0 & a_1 \\ \mp a_1 c & a_0 \end{bmatrix}$ then
$M \cdot \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ \mp a_1 c & a_0 \end{bmatrix} \cdot \begin{bmatrix} b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Coefficients in matrix M may be taken from general form of element $C = A \cdot B$.

We will make transformation: $\begin{bmatrix} b_1 \\ b_0 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 \\ \mp a_1 c & a_0 \end{bmatrix}^{-1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = M^{-1} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix}$

Now we can do as follow: the determinant of matrix M is equal to:
$det(M) = a_0^2 \pm a_1^2 c$
Then:
$M^{-1} = \frac{1}{\det(M)} \begin{bmatrix} a_0 & -a_1 \\ \pm a_1 c & a_0 \end{bmatrix}$
And
$\begin{bmatrix} b_1 \\ b_0 \end{bmatrix} = \frac{1}{\det(M)} \begin{bmatrix} a_0 & -a_1 \\ \pm a_1 c & a_0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \frac{1}{\det(M)} \begin{bmatrix} -a_1 \\ a_0 \end{bmatrix}$

We can compute it making substitutions:

1. $D = a_0^2$
2. $E = a_1^2 c$
3. $H = E + D = \det(M)$
4. $\overline{H} = H^{-1}$
5. $b_1 = -a_1 \overline{H}$
6. $b_0 = a_0 \overline{H}$

Of course to count inversion in F_{p^2} we need to count one inversion, 4 multiplications, 1 multiplication by small constant c and 1 addition in F_p.

3 GLV and GLS methods

Elliptic curve cryptosystems are very popular nowadays. The most important operations are point multiplication $[k]P$ and multiexponentiation $[k]P + [l]Q$. There are used different methods to compute $[k]P$ when k vary and P is fixed and different when both k and P vary.

Unfortunately decomposition is not so easy in the case when both n and P vary. Fortunately for some types of elliptic curves there is possibility to make decomposition of $[k]P$ into $[k]P + [k_1]P_1$ using endomorphism. Because we want to speed-up the computations, the chosen endomorphism should be fast computable.

In [5] Galbraith, Lin and Scott found out that this method may be used to speed-up computations on many elliptic curves over F_{p^2}. Elliptic curves over F_{p^2} are less vulnerable for index calculus attacks than curves defined over F_{p^m} for $m > 2$.

Unfortunately operations in field extensions F_{p^m} are much more computationally harder (and longer) than the same operations in F_p, especially inversion and multiplication. It is hard to avoid this problem in software solutions but in hardware we are able to make multiplication and inversion in F_{p^2} in almost the same time like in F_p.

In [9] it is described how to make hardware multiplication in F_{p^2} using RNS (Residue Number System) method. We propose an alternative method of hardware implementation of operations in F_{p^2} using parallel computations. It is generic method so any method of multiplication (for example RNS) may be applied. Moreover, using this method we can make point scalar multiplication $2 - 2.6$ times faster comparing to methods using standard multiplication in F_{p^2} and $5 - 9$ times faster comparing to classic solutions for elliptic curves over F_h, where h is prime number about twice longer than p. It is important that curves over F_{p^2} and curves over F_h have then similar level of security.

4 Edwards curves and twisted Edwards curves

Edwards and twisted Edwards curves are described with many additional details in [3], [4] and [13]. Below we present only the most important information about Edwards and twisted Edwards curves.

4.1 Edwards and twisted Edwards curves

Twisted Edwards curve over field K with characteristic not equal 2 is given by formula:

$E_t : ax^2 + y^2 = 1 + dx^2y^2$, where $ad(a - d) \neq 0$

For every twisted Edwards curve exists birationally equivalent short Weierstrass curve but not for every short Weierstrass curve exists birationally equivalent twisted Edwards curve. The sum of two points in affine coordinates $(x_1, y_1), (x_2, y_2)$ on curve E_t is:

$P + Q = (\frac{x_1y_2+y_1x_2}{1+dx_1x_2y_1y_2}, \frac{y_1y_2-ax_1x_2}{1-dx_1x_2y_1y_2})$

These formula is complete (or unified) if a is square and d is non-square in K.

Edwards curve has $a = 1$ and $d \neq \{0, 1\}$. For such curve it is easy to see that point $(0, 1)$ is the neutral element of addition law. Points $(1, 0)$ and $(-1, 0)$ have order 4 and point $(0, -1)$ has order 2. Point $(-x, y)$ is negative to point (x, y). Moreover, the presented addition law is unified: it can be used to double a point and works also for neutral element. If d is nonsquare in K then addition law is complete (works for all pairs of inputs). Using Edwards addition law (especially using inverted coordinates) requires much less multiplications than standard coordinates systems on short Weierstrass curve (like projective coordinates).

Points addition on twisted Edwards curve in inverted coordinates requires 12 multiplications (9 multiplications, 1 squaring and 2 multiplications by constants) and 7 additions/subtractions. Sometimes we can assume that $Z_1 = 1$. Then algorithm requires 1 multiplication less.

Point doubling requires 9 multiplications (3 multiplications, 4 squares, 2 multiplications by constants) and 6 additions/subtractions.

Moreover, in both algorithms if we put $a = 1$ then we obtain arithmetic on Edwards curve and algorithm for point doubling and algorithm for points addition require one multiplication less.

On the other hand, on short Weierstrass curve in projective coordinates we need to compute for points addition 14 multiplications (12 multiplications and 2 squares) and 7 additions/subtractions (or 6 additions and one multiplication by 2).
The point doubling on short Weierstrass curve requires 12 multiplications (5 multiplications, 6 squares and 1 multiplication by constant) and 12 additions/subtractions (7 additions/subtractions, 3 multiplications by 2 and 1 multiplication by 3).

Moreover, on Edwards and twisted Edwards curves we can use unified arithmetic (use addition formula both for addition and doubling) which is not possible on short Weierstrass curve.

4.2 Endomorphism on twisted Edwards curves

Theorem. ([12], theorem 3, p. 8)

Let $char(F_p) > 3$ be prime number and let $E_{E,a,d}$ be a twisted Edwards curve over F_p with $p + 1 - t$ points. Let $E^t_{E,a,d}$ over F_{p^2} be the quadratic twist of $E_{E,a,d}(F_{p^2})$, then $\#E^t_{E,a,d}(F_{p^2}) = (p-1)^2+t^2$. Let $r|\#E^t_{E,a,d}(F_{p^2})$ be a prime number such that $r > 2q$. Let $\phi : E_{E,a,d} \to E^t_{E,a,d}$ be the twisting isomorphism defined over F_{p^4}. Let:

$\pi^t_p = \phi \circ \pi \circ \phi^{-1}$.

For $P \in E^t_{E,a,d}(F_{p^2})[r]$, we have $(\pi^t_p)^2(P) + P = O$.

Proof:

by well-known Weil theorem, we have $E_{E,a,d}(F_{p^2}) = (p + 1)^2 - t^2$ and $E^t_{E,a,d}(F_{p^2}) = (p - 1)^2 + t^2$. Since $r > 2p$, hence $r \nmid \#E_{E,a,d}(F_{p^2}) = (p + 1 - t)(p + 1 + t)$. Therefore by the assumption of the theorem, one have

$r|\#E^t_{E,a,d}(F_{p^2}) = \#E^t_{E,a,d}(F_{p^2})\#E_{E,a,d}(F_{p^2})$ while $r \nmid \#E^t_{E,a,d}(F_{p^4})$. This implies that for $P \in E^t_{E,a,d}(F_{p^2})[r]$, $\pi^t_p(P)$ belongs to $E^t_{E,a,d}(F_{p^2})[r]$. It follows that for $P \in E^t_{E,a,d}(F_{p^2})[r]$ there exists $\lambda \in Z$, such that $\pi^t_p(P) = [\lambda]P$.

By the definition as above, $\pi^t_p(x,y) = \left(\alpha^{\frac{p-1}{2}}x^p, y^p\right)$, where $\alpha \in F_{p^2}$ is not a square in F_{p^2}. And hence:

$\left(\pi^t_p\right)^2(x,y) = \left(\alpha^{\frac{p^2-1}{2}}x^{p^2}, y^{p^2}\right)$. Since $\alpha \in F_{p^2}$ is not square in F_{p^2}, so $\alpha^{\frac{p^2-1}{2}} = -1$. By the assumption of the theorem, $P \in E^t_{E,a,d}(F_{p^2})$, we have $x^{p^2} = x$, $y^{p^2} = y$. Therefore,

$\left(\pi^t_p\right)^2(x,y) = (-x,y)$.

Using corollary 1 p. 5 from [5] we get that $\lambda^2 + 1 \equiv 0 (\mathrm{mod}\, r)$, so we can easy precompute λ.

4.3 Choosing proper curve and its parameters

It is easy to see that if we want to use GLV-GLS method we have to carefully choose parameters of our twisted Edwards curve. We know that $p \equiv 1 (\mathrm{mod}\, 4)$. Moreover, because equation $\lambda^2 + 1 = 0 (\mathrm{mod}\, r)$ must have solution, it means that $r \equiv 1 (\mathrm{mod}\, 4)$.

Because of these all assumptions it is easy to see that irreducible polynomial which will generates F_{p^2} cannot be of form $x^2 + 1$. It may be of the form:

1. $x^2 + 2$ if $p \equiv 5 (\mathrm{mod}\, 8)$
2. $x^2 \pm c$, $c \neq 1, 2$ if $p \equiv 1 (\mathrm{mod}\, 8)$

In the second case we will be supposed to use Karatsuba multiplication method in F_{p^2}. In the first case we need to choose best solution for our application.

We give our proposition of algorithm for searching twisted Edwards curves over F_{p^2} which are proper for cryptographic solutions.

Algorithm: Generation of twisted Edwards curve parameters:
Input: Prime p for which $p \equiv 1 (\mathrm{mod}\, 4)$

1. Find the smallest $|d|$ for which:
   (a) $d \neq \{0, 1\}$
   (b) d is not-square in K
   (c) $r = \frac{\#E_t}{4}$ is prime and $r \equiv 1 (\mathrm{mod}\, 4)$, where $E^t_{E,\sqrt{d},\sqrt{d^3}} : \sqrt{d}x^2 + y^2 = 1 + \sqrt{d^3}x^2y^2$ over F_{p^2}
   (d) $\lambda = \sqrt{-1} (\mathrm{mod}\, r)$ and ($\lambda = \lfloor\sqrt{r}\rfloor$ or $r - \lambda = \lfloor\sqrt{r}\rfloor$)
2. Find $G \in E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})$ for which $\#G = r$

3. $\alpha = d^{\frac{1-p}{4}}$

4. if $\lambda = r - \lfloor\sqrt{r}\rfloor$

5. $\quad\alpha = -\alpha$

6. end if

7. if $E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})$ is not cryptographically secure go to 1.

Output: Parameters for twisted Edwards curve given by equation:$E^t_{E,a,d}$: $a_t x^2 + y^2 = 1 + d_t x^2 y^2$ with parameters:
$a_t = \sqrt{d}$
$d_t = \sqrt{d^3}$
$r = \frac{\#E_t}{4}$
$G = (x_t, y_t)$ for which $Ord(G) = r$
$\lambda = r - \lfloor\sqrt{r}\rfloor$ or $\lambda = \lfloor\sqrt{r}\rfloor$
$a = d^{\frac{1-p}{4}}$

Note that if $\lambda = \lfloor\sqrt{r}\rfloor$ then $[\lambda]P = (\alpha x^p, y^p)$ and if $\lambda = r - \lfloor\sqrt{r}\rfloor$ then $[\lambda]P = -(\alpha x^p, y^p) = (-\alpha x^p, y^p)$

Example 1.

Let $p = 1048589$ be the prime number. We are looking for twisted Edwards curve over F_{p^2} which is given by equation (see [12] p.9). We have Edwards curve given by equation:

$E_{E,1,d} : x^2 + y^2 = 1 + dx^2y^2$ and twisted Edwards curve given by equation $E^t_{E,\sqrt{d},\sqrt{d^3}} : \sqrt{d}x^2 + y^2 = 1 + \sqrt{d^3}x^2y^2$

We choose $d = 229$ which is not-square in F_p. Then of course d is square in F_{p^2} and $\sqrt{d} = 361057t$ and $\sqrt{d^3} = 892111t$.

$\#E_t$ is equal to 1099536793748. Then E_t has torsion subgroups $Z/2Z \times Z/549\,768\,396874Z$. Because number 549 768 396874 is not prime, then we need to find generator G' of order equal to 274 884 198 437, because it is the biggest prime order of any point on this curve. We can to do this finding firstly generator G of torsion subgroups $Z/549\,768\,396874$. Then $G' = [2]G$ has order 274 884 198 437. We should note, that Edwards and twisted Edwards curves always have order of points group divisible by 4. Because $r = Ord(G') = 274\,884\,198\,437 > 2p$ then twist Frobenius map on $E^t_{E,\sqrt{d},\sqrt{d^3}}$ may be written as $\pi^t_p(x, y) = (d^{\frac{1-p}{4}}x^p, y^p)$. Then for every $P \in E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})$ there is $\left(\pi^t_p\right)^2(x, y) = (d^{\frac{1-p^2}{4}}x^{p^2}, y^{p^2})$. If d is not square in F_p and $p \equiv 1 (\mathrm{mod}\, 4)$, then $d^{\frac{1-p^2}{4}} = -1$ in F_{p^2}. It means that if $\pi^t_p(x, y) = (d^{\frac{1-p}{4}}x^p, y^p) = [\lambda]P$, then $\left(\pi^t_p\right)^2(P) + P = O_{E_t}$ and $\lambda^2 + 1 \equiv 0 (\mathrm{mod}\, r)$.

Then:
$G' = (620046t + 978259, 716864t + 443811)$
$\alpha = d^{\frac{1-p}{4}} = 1009596$
$\lambda = 524294$
One can note that $\lfloor\sqrt{r}\rfloor = 524294 = \lambda$. It means that λ is chosen optimal.

4.4 Computing of endomorphism

We should see that we can count the endomorphism very fast. Let's look that if $x = x_1t + x_0$ and $y = y_1t + y_0$ then $x^p = (x_1t + x_0)^p = x_1^p t^p + x_0^p = x_1t^p + x_0$. Because we make all computations modulo $F(t) = t^2 \pm c$ then we can easy find out that for every $k \in F_p$ there is $t^k = (\mp c)^{\lfloor \frac{k}{2} \rfloor} t^{k \bmod 2}$ over F_{p^2}. Then for $k = p$ we get $t^p = (\mp c)^{\lfloor \frac{p}{2} \rfloor} t^{p \bmod 2} = -t$. So finally $x^p = x_1t^p + x_0 = -x_1t + x_0$. As same: $y^p = -y_1t + y_0$ and $[\lambda]P = [\lambda](x, y) = [\lambda](x_1t + x_0, y_1t + y_0) = (\alpha x^p, y^p) = (\alpha(-x_1t + x_0), -y_1t + y_0)$. Finally:

$[\lambda]P = (\alpha x^p, y^p) = (-\alpha x_1t + \alpha x_0, -y_1t + y_0)$.

This endomorphism is very easy to compute and requires only two multiplications in F_p.

Because we want to use this endomorphism to make decomposition of $[k]P$ into $[k_1]([\lambda]P) + [k_0]P$ it would the best, if k_1 and k_0 would be of the same bitlength. If we choose such λ that $\lambda^2 + 1 = r$, then $\lambda = \lfloor \sqrt{r} \rfloor$ and finally k_1 and k_0 will have the same bitlength (in average case).

The decomposition will be given then by formula $[k]P = \left\lfloor \frac{k}{\lambda} \right\rfloor ([\lambda]P) + [k \bmod \lambda]P = \left\lfloor \frac{k}{\lambda} \right\rfloor P_1 + [k \bmod \lambda]P$.

In FPGA such decomposition will require additional component which would be used to count both $\left\lfloor \frac{k}{\lambda} \right\rfloor$ and $[k \bmod \lambda]$. Although it is not so much complicated it will require additional resources and time.

4.5 $\lambda^2 + 1 = r$. Is it possible?

Our aim is to count both $\left\lfloor \frac{k}{\lambda} \right\rfloor$ and $[k \bmod \lambda]$ very fast. We can see that it would be possible if $\lambda = 2^s$. Then $\left\lfloor \frac{k}{2^s} \right\rfloor$ we can count just shifting k for s position right and $[k \bmod 2^s]$ would be given by s least significant bits of number k.

Because $\lambda^2 + 1 \equiv 0 (\bmod\, r)$ then if $\lambda = 2^s$ then $(2^s)^2 + 1 = 2^{2s} + 1 \equiv 0 (\bmod\, r)$.

If it is not necessary that both k_1 and k_0 are of the same bitlength, then we can, for example, to search for $\lambda \in [\frac{\lfloor \sqrt{r} \rfloor}{4}; 4 \lfloor \sqrt{r} \rfloor]$. However, we should remember then that one of k_1, k_0 will be longer than $\frac{n}{2}$ bits (of course if $\lambda^2 + 1 \neq r$) and then, even if we will count $[k_1]P_1$ and $[k_0]P_0$, even using parallel computations one of these computations will require more time. That is why we would like to find such λ that $\lambda^2 + 1 = r$.

It is easy to see that if $\lambda^2 + 1 = 2^{2s} + 1 = r$ (where r is prime number) it implies that $2s = 2^h$ and then $2^{2s} = 2^{2^h} + 1 = r$ must be Fermat prime number. Fermat number F_n is given by formula $2^{2^n} + 1$.

Unfortunately, there are only five such numbers F_0, F_1, F_2, F_3 and F_4 which are known to be prime. Moreover, even if there are some other Fermat prime numbers they would be to big to use them in real applications and F_0, F_1, F_2, F_3 and F_4 are to small.

Instead of this fact, let's look that if $\lambda = 2^{2^{h-1}}$ and $\lambda^2 + 1 = 2^{2^h} + 1 = r$ and r is Fermat prime number then k_1 and k_0 are of the same bitlength and we would be able to compute k_1 making shifting k and k_2 getting the least significant bits of number k. It means that we would not have to make additional component

which would require many resources and we would save some time, because shifting is very fast operation.

We should also remember that these values we may get only for some primes p. We should remember that:

$\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2}) = (p-1)^2 + t^2$, where $t \in \{-2\sqrt{p}+1, ..., 2\sqrt{p}+1\}$.

It means, that:

$(p-1)^2 \leq \#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2}) = 4r \leq (p-1)^2 + (2\sqrt{p}+1)^2 = p^2 - 2p + 1 + 4p + 4\sqrt{p} + 1 = p^2 + 2p + 4\sqrt{p} + 2$

Unfortunately, that there are only (at most) three integers satisfying this formula:

Let's denote by $p_{\max}$ the biggest value for which formula is true, which means that:

$(p_{\max}-1)^2 \leq \#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2_{\max}})$ so $p_{\max} \leq \sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2_{\max}})} + 1$.

Let's denote by $p_{\min}$ the smallest value for which formula is true. Then:

$\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2_{\min}}) \leq p^2_{\min} + 2p_{\min} + 4\sqrt{p_{\min}} + 2 < (p_{\min}+2)^2 = p^2_{\min} + 4p_{\min} + 4$, so $p_{\min} > \sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2_{\min}})} - 2$ and finally $p_{\min} \geq \sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2_{\min}})} - 1$

So $p \in \{\sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})} - 1, \sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})}, \sqrt{\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2})} + 1\}$.

The only curve we found with the smallest possible $\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2}) = 4r$ was curve with $p = 5$ and $\lambda = 2^{2^0} = 2$. $\#E^t_{E,\sqrt{d},\sqrt{d^3}}(F_{p^2}) = 20 = 5 \cdot 4$. Of course this curve is absolutely useless in real applications.

5 GLV-GLS method for twisted Edwards curves over F_{p^2} in hardware implementations

Presented solution using parallel computations to speed-up multiplication in F_{p^2} seems to be very useful in hardware implementations. There is work [9] which presents FPGA implementation of GLV method but it uses RNS method. RNS method is very vulnerable for DPS blocks which we have in device. Moreover, in our solution may be used any method of multiplication in F_p if only in device will be enough resources for its implementation.

6 Speed-up

Our method may be used to speed up, especially hardware, implementations. Comparing to others implementations, when operations in F_{p^2} were not made parallel our solution may up to three time faster.

We know that point addition on twisted Edwards curve requires 12 multiplications (8 multiplications, 1 squaring and 2 multiplications by constants) and 7

additions/subtractions. Point doubling requires 9 multiplications (3 multiplications, 4 squares, 2 multiplications by constants) and 6 additions/subtractions. We assume that addition requires one processor cycle.

Let T_M be the number of processor cycles required for multiplication in F_p. Then it is easy to count the number of processor cycles for other operation. In table below we present comparison between our solution, solution with standard method of multiplication in F_{p^2} and classic solution of elliptic curve in short Weierstrass form over F_h where h is about twice longer than p. We consider inversion method using fast exponentiation, because this method is easy to implement in hardware and not need many additional resources. Moreover, inversion is counted only once, at the end of point scalar multiplication. We consider that we compute inversion by fast exponentiation method (FE).

Operation	Ours $p \equiv 1 (\mathrm{mod}\, 8)$
Multiplication in F_p	$T_M + \lceil \log_2 c \rceil + 1$
Inversion in F_p using FE	$\frac{3}{2} \lceil \log_2 (p-2) \rceil T_M \approx \frac{3}{2} n T_M$
Inversion in F_{p^2}	$T_M \left(\frac{3}{2} n + 4\right) + \lceil \log_2 c \rceil + 1$
Points addition	$12 T_M + 12 \lceil \log_2 c \rceil + 19$
Point doubling	$9 T_M + 9 \lceil \log_2 c \rceil + 15$
multiplication ununified	$n \left(7.5 T_M + 7.5 \lceil \log_2 c \rceil + 12.25\right)$
multiplication ununified with inversion	$T_M \left(9n + 4\right) + \lceil \log_2 c \rceil \left(1 + \frac{15}{2} n\right) + +1 + \frac{49}{4} n$
Multiplication unified	$n \left(9 T_M + 9 \lceil \log_2 c \rceil + 14.25\right)$
multiplication unified with inversion	$T_M \left(4 + 10.5n\right) + (9n + 1) \lceil \log_2 c \rceil + +14.25n + 1$

Table 1. Number of processor cycles for operations in F_{p^2} using our method for $p \equiv 1 (\mathrm{mod}\, 8)$

Operation	Ours $p \equiv 5 (\mathrm{mod}\, 8)$
Multiplication in F_p	$T_M + 2$
Inversion in F_p using FE	$\frac{3}{2} \lceil \log_2 (p-2) \rceil T_M \approx \frac{3}{2} n T_M$
Inversion in F_{p^2}	$\left(\frac{3}{2} n + 4\right) T_M + 2$
Points addition	$12 T_M + 31$
Point doubling	$9 T_M + 24$
multiplication ununified	$n \left(7.5 T_M + 19.75\right)$
multiplication ununified with inversion	$\left(4 + 9n\right) T_M + 19.75n + 2$
Multiplication unified	$n \left(9 T_M + 23.25\right)$
multiplication unified with inversion	$\left(4 + 10.5n\right) T_M + 23.25n + 2$

Table 2. Number of processor cycles for operations in F_{p^2} using our method for $p \equiv 5 (\mathrm{mod}\, 8)$

Operation	Standard
Multiplication in F_p	$3T_M + 4$
Inversion in F_p using FE	$\frac{3}{2}\lceil \log_2(p-2) \rceil T_M \approx \frac{3}{2} n T_M$
Inversion in F_{p^2}	$\left(\frac{3}{2}n + 4\right) T_M + 2$
Points addition	$36T_M + 55$
Point doubling	$27T_M + 42$
Multiplication ununified	$n\,(22.5T_M + 34.75)$
Multiplication ununified with inversion	$(4 + 24n)\,T_M + 34.75n + 2$
Multiplication unified	$n\,(27T_M + 41.25)$
Multiplication unified with inversion	$(4 + 28.5n)\,T_M + 41.25n + 2$

Table 3. Number of processor cycles for operations in F_{p^2} using standard multiplication method

Operation	Classic
Multiplication in F_p	$2T_M$ (because $\log_2 h \approx 2\log_2 p$)
Inversion in F_h using FE	$3\lceil \log_2(h-2) \rceil T_M \approx 6nT_M$
Points addition	$28T_M + 14$
Point doubling	$24T_M + 24$
Multiplication ununified	$n\,(76T_M + 62)$
Multiplication ununified with inversion	$n\,(82T_M + 62)$
Multiplication unified	impossible
multiplication unified with inversion	impossible

Table 4. Number of processor cycles for operations in F_h using classic method ($\log_2 h \approx 2\log_2 p = 2n$)

Method	cycles	% of classic method
Ours $p \equiv 1 (\mathrm{mod}\, 8)$ ununified	$T_M\,(9n + 4) + + \lceil \log_2 c \rceil \left(1 + \frac{15}{2}n\right) + 1 + \frac{49}{4}n$	Depends on $\lvert c \rvert$
Ours $p \equiv 1 (\mathrm{mod}\, 8)$ unified	$T_M\,(4 + 10.5n) + +(9n + 1)\lceil \log_2 c \rceil + 14.25n + 1$	Depends on $\lvert c \rvert$
Ours $p \equiv 5 (\mathrm{mod}\, 8)$ ununified	$(4 + 9n)\,T_M + 19.75n + 2$	$11.0\% \leq N_c \leq 20.0\%$
Ours $p \equiv 5 (\mathrm{mod}\, 8)$ unified	$(4 + 10.5n)\,T_M + 23.25n + 2$	$12.8\% \leq N_c \leq 23.4\%$
Standard ununified	$(4 + 24n)\,T_M + 34.75n + 2$	$29.3\% \leq N_c \leq 48.4\%$
Standard unified	$(4 + 28.5n)\,T_M + 41.25n + 2$	$34.8\% \leq N_c \leq 48.4\%$
Classic	$n\,(79T_M + 62)$	$N_c = 100\%$

Table 5. Comparison of mutliplication methods with inversion at the end of computations

7 Twisted and generalized Hessian curves

7.1 Generalized Hessian curve

Generalize Hessian curves are described in [18]. Generalized Hessian curve $E_{G,c,d}$ over field F is given by equation:

In affine coordinates $E_{G,c,d} : x^3 + y^3 + c = dxy$

or by

$E_{G,c,d} : X^3 + Y^3 + cZ^3 = dXYZ$ in projective coordinates. There is requirement that $c\left(27c - d^3\right) \neq 0$.

If $c = 1$ then $E_{G,1,d} : x^3 + y^3 + 1 = dxy$ is Hessian curve.

The inversion of $P = (x_1, y_1)$ is $-P = (y_1, x_1)$ in affine coordinates

In projective coordinates neutral element is $(1, -1, 0)$ and inversion of $P = (X_1, Y_1, Z_1)$ is point $-P = (Y_1, X_1, Z_1)$. We can use on generalized Hessian curve complete addition formula over F_q for example always when $q \equiv 1 (\mathrm{mod}\, 3)$ and c is not cube in F_q.

Addition formula on generalized Hessian curve requires 13 multiplications (12 multiplications by vary elements and 1 multiplication by constant) and 3 additions/subtractions.

Doubling formula requires (for $p \neq 2$) takes 9 multiplications (7 multiplications by vary elements, 1 square and 1 multiplication by constant) and 14 additions/subtractions.

7.2 Twisted Hessian curves

Twisted Hessian curves are described in [16] by Bernstein et. all.

Twisted Hessian curve over field K is given by equation:

$E_{H,a,d} : ax^3 + y^3 + 1 = dxy$ with special point $(0, -1)$ in affine coordinates or:

$aX^3 + Y^3 + Z^3 = dXYZ$ in projesctive coordinates with special point $(0, -1, 1)$.

Elements $a, d \in K$ and $a(27a - d^3) \neq 0$.

If $a = 1$ then $E_{H,a,d} : ax^3 + y^3 + 1 = dxy$ is Hessian curve

On twisted Hessian curve we can use fast arithmetic. Moreover, we can use complete formula (the same formula for all operations: points addition, point doubling etc.) on twisted Hessian curve over F_q if $q \equiv 1 (\mathrm{mod}\, 3)$ and a i not cube in F_q.

Arithmetic on twisted Hessian curves is described with all details in [16].

Best complete addition formula requires 12 multiplications (11 multiplications of vary elements and 1 multiplication by constant) and 17 additions/subtractions.

We can also use complete addition formula which requires 13 multiplications (12 multiplications of vary elements and 1 multiplication by constant) and 3 additions/subtractions.

Because we assume that in F_q multiplication requires the same time as square, we can use different formulas for doubling if we need not complete formula.

We can choose doubling with cost of 9 multiplications (7 multiplications of vary elements, 1 square, 1 multiplication by constant) and 11 additions (we assume that for doubling and tripling we use addition of elements). This formulas may be found in [16], p. 19-20. There is requirement that $2 \neq 0$ in the field.

However generalized Hessian curves and twisted Hessian curves are isomorphic, on twisted Hessian curves we can use a little bit faster arithmetic.

7.3 Isomorphism between twisted Hessian curves and elliptic curves in short Weierstrass form over finite fields

Let's consider that we want to compute on elliptic curve $E_{W,A,B}(F_q) : y^2 = x^3 + Ax + B$ and generator $P_{W,A,B}$ point $Q_{W,A,B} = [k]P_{W,A,B}$ in short Weierstrass form we can find isomorphic twisted Hessian curve if and only if $3|\#E_{W,A,B}(F_q)$ and $q \equiv 1 (\mathrm{mod}\, 3)$, what means that there is point of order 3 on $E_{W,A,B}(F_q)$. Of course none of elliptic curves over F_p for which $\#E_{W,A,B}(F_p)$ is prime has isomorphic twisted Hessian curve over F_p.

It is easy to see that for some elliptic curves over F_p for which $\#E_{W,A,B}(F_p)$ is prime we will be able to find for $\#E_{W,A,B}(F_{p^2})$ subgroup isomorphic to this curve on twisted Hessian curve over F_{p^2}.

Let's see, thet if $\#E_{W,A,B}(F_p) = p + 1 - t$ over F_p then $\#E_{W,A,B}(F_{p^2}) = (p+1)^2 - t^2$ over F_{p^2}. It means that there is possibility that $\#E_{W,A,B}\left(F_{p^2}\right)$ is divisible by 3. For large p of course $p^2 \equiv 1 (\mathrm{mod}\, 3)$. It means that if $3|\#E_{W,A,B}(F_{p^2})$, then we can make operations on twisted Hessian curve over F_{p^2} instead of short Weierstrass curve over F_p.

We checked this property for NIST elliptic curves over F_p. We found that for NIST P-224 and NIST P-256 we can find twisted Hessian curve over F_{p^2} isomorphic to $E_{W,A,B}(F_{p^2})$.

Now we will show how to find such twisted Hessian curve:

Let's suppose that we have given triangular elliptic curve $E_{TR,a,d} : \overline{y}^2 = d\overline{x}\overline{y} + a\overline{y} = \overline{x}^3$ over F_{p^2}, where $a, d \in F_p$. We can make transformations:

$(\overline{y} + \frac{dx+a}{2})^2 = \left(\overline{x} + \frac{d^2}{12}\right)^3 + \left(\frac{da}{2} - \frac{d^4}{48}\right)\left(\overline{x} + \frac{d^2}{12}\right) - \frac{d^2}{12}\left(\frac{da}{2} - \frac{d^4}{48}\right) + a^2$. If:

$E_{W,A,B} : y^2 = x^3 + Ax + B$ then:

$x = \overline{x} + \frac{d^2}{12}$

$y = \overline{y} + \frac{dx+a}{2}$

$A = \frac{da}{2} - \frac{d^4}{48}$

$B = -\frac{d^2}{12}A + a^2$

For elliptic curves over F_p we can extend field form F_p to F_{p^2}. Then of course coefficients of such curve over field extension stil belong to F_p. Then If we know $A, B \in F_p$ and we want to find out $a, d \in F_{p^2}$ we need to make computations below:

d is one of roots of polynomial:

$W(s) = \frac{-1}{6912}s^8 - \frac{1}{24}As^4 - Bs^2 + A^2$

then $a = (A + \frac{d^4}{48})\frac{2}{d}$

It is easy to see that in projective coordinates $E_{TR,a,d} : VW\,(V + dU + aW) = U^3$:

Then for triangular curve $E_{TR,a,b}$ we can easy find out isomorphic twisted Hessian curve:

$E_{H,(d^3-27a),3d} : (d^3 - 27a)X^3 + Y^3 + Z^3 = 3dXYZ$

and

$X = U$

$Y = \omega(V + dU + aW) - \omega^2 V - aW$

$Z = \omega^2\,(V + dU + aW) - \omega V - aW$

where ω is not trivial cubic root from 1 and $X, Y, Z, \omega \in F_{p^2}$

Now we can use fast arithmetic (or complete arithmetic if someone wants to have solution invoulnerable for side channel attacks) to count $Q_{H,(d^3-27a),3d} = [k]P_{H,(d^3-27a),3d}$ on isomorphic twisted Hessian curve instead of on short Weierstrass curve.

After computations of $Q_{H,(d^3-27a),3d} = (X_Q, Y_Q, Z_Q)$ we need to back on $E_{W,A,B}$ over F_p to point on $Q_{W,A,B} = (x_Q, x_Q)$ (in affine coordinates) on $E_{W,A,B}$ over F_p.

Firstly we find point on triangular curve $Q_{TR,a,d} = (U_Q, V_Q, W_Q)$:

$U_Q = X_Q$

$V_Q = -(dX_Q + \omega Y_Q + \omega^2 Z_Q)/3$

$W_Q = -(dX_Q + Y_Q + Z_Q)/(3a)$

Finally:

$x_Q = \frac{U_Q}{W_Q} + \frac{d^2}{12}$

$y_Q = \frac{V_Q}{W_Q} + \frac{dx+a}{2}$

So $Q_{W,A,B} = (x_Q, y_Q) = [k]P_{W,A,B}$ is the result. Now we have point on $E_{W,A,B}$ and of course $x_Q, y_Q \in F_p$.

7.4 Speed-up for NIST curves

Using presented ideas we are able to speed-up point scalar multiplication on two NIST curves over F_p. These curves are NIST P-224 and NIST P-256. For others NIST curves over large prime fields the smallest field extension to get isomorphic twisted Hessian curve is:

- 8 for NIST P-192 and NIST P-384
- 4 for NIST P-521

Because we showed in [17] that for elliptic curves for which order of points group is prime number we can always find 2-isogenous twisted Edwards curve over F_{p^3} we will be interested in isomorphic twisted Hessian curves only if degree of field extension will be at most 2.

7.4.1 NIST P-224

For NIST P-224 we can find twisted Hessian curve over F_{p^2} which has subgroup isomorphic to NIST P-224.

We can use irreducible polynomial of form $F(t) = t^2 + 11$ for arithmetic in F_{p^2}. Multiplication using such polynomial requires then $T_M + \lceil \log_2 11 \rceil + 1 = T_M + 5$ processor cycles, where T_M is number of processor cycles required for multiplication in F_p.

7.4.2 NIST P-256

For NIST P-224 we can find twisted Hessian curve over F_{p^2} which has subgroup isomorphic to NIST P-224.

We can use irreducible polynomial of form $F(t) = t^2 + 1$ for arithmetic in F_{p^2}. Multiplication using such polynomial requires then $T_M + 2$ processor cycles, where T_M is number of processor cycles required for multiplication in F_p.

7.5 Comparison with other methods of point scalar multiplication

In [17] we presented fast method of point scalar multiplication on all NIST curves over F_p using (twisted) Edwards curves over F_{p^3}. Using this solution it is possible to use complete formula to protect device against side channel attacks. Unfortunately, arithmetic in F_{p^3} is much more complicated. We are able to do this arithmetic only a little bit slower than arithmetic in F_p but such hardware solutions require many resources.

In this case arithmetic on twisted Hessian curves may be very interesting, because:

a) It is faster than classic arithmetic on NIST curves in short Weierstrass form over F_p in hardware

b) It allows us to use complete formula as solution presented in [17]

c) It requires much less resources than solution from [17] but still more than for short Weierstrass form over F_p (we estimate that using twisted Hessian curves over F_{p^2} requires from two to three times more resources)

We show comparison of this three solutions below for each of curves (NIST P-224 and NIST P-256):

Curve	tHc over F_{p^2}	SWC over F_p
Multiplication	$T_M + 5$	T_M
Inversion in F_p using FE	$\frac{3}{2}\lceil \log_2(p-2) \rceil T_M \approx 336T_M$	$336T_M$
Inversion in F_{p^2}	$340T_M + 1706$	-
Points addition	$12T_M + 77$	$14T_M + 7$
Point doubling	$9T_M + 56$	$12T_M + 12$
multiplication ununified	$3248T_M + 21\,168$	$4256T_M + 3472$
multiplication ununified with inversion	$3588T_M + 22\,874$	$4592T_M + 3472$
Multiplication unified	$4032T_M + 25\,872$	$5824T_M + 4256$
multiplication unified with inversion	$4372T_M + 27578$	$6160T_M + 4256$

Table 6. Comparison of number of processor cycles for twisted Hessian curve (tHc) over F_{p^2} and short Weierstrass curve (SWC) over F_p for NIST P-224

Curve	tHc over F_{p^2}	SWC over F_p
Multiplication	$T_M + 2$	T_M
Inversion in F_p using FE	$\frac{3}{2}\lceil \log_2(p-2) \rceil T_M \approx 384T_M$	$384T_M$
Inversion in F_{p^2}	$388T_M + 390$	-
Points addition	$12T_M + 29$	$14T_M + 7$
Point doubling	$9T_M + 20$	$12T_M + 12$
multiplication ununified	$3712T_M + 8832$	$4864T_M + 3968$
multiplication ununified with inversion	$4100T_M + 9222$	$5248T_M + 3968$
Multiplication unified/ladder	$4608T_M + 11\,136$	$6656T_M + 4864$
multiplication unified/ladder with inversion	$4996T_M + 11\,526$	$7040T_M + 4864$

Table 7. Comparison of number of processor cycles for twisted Hessian curve (tHc) over F_{p^2} and short Weierstrass curve (SWC) over F_p for NIST P-256

N_{pc} is number of processor cycles needed for operation.

Method	cycles	% of SWC method
tHc over F_{p^2} NIST P-224, ununified	$3588T_M + 22874$	$80.09\%N_{pc_1} \leq$
SWC over F_p NIST P-224	$4592T_M + 3472$	N_{pc_1}
tHc over F_{p^2} NIST P-224, unified	$4372T_M + 27578$	$72.75\%N_{pc_2} \leq$
SWC over F_p NIST P-224, ladder	$6160T_M + 4256$	N_{pc_2}
tHc over F_{p^2} NIST P-256, ununified	$4100T_M + 9222$	$78.58\%N_{pc_3}$
SWC over F_p NIST P-256	$5248T_M + 3968$	N_{pc_3}
tHc over F_{p^2} NIST P-256, unified	$4996T_M + 11\,526$	$71.41\%N_{pc_4} \leq$
SWC over F_p NIST P-256, ladder	$7040T_M + 4864$	N_{pc_4}

Table 8. Comparison of mutliplication methods with inversion at the end of computations

We also made assumptions that to get device involnurable for side channel attack, we can use Montgomery ladder for point scalar multiplication on short Weierstrass curve over F_p.

It is not good idea to use twisted Hessian curves over F_{p^2} in situations when multiplication is very short, because we have to do then many additions (in worst case, when multiplication takes 1 processor cycle, it is almost 3.3 times slower for NIST P-224 and almost 1.5 times for NIST P-256).

Finally, if multiplication is not extremely short, we can make point scalar multiplication much faster using twisted Hessian curves over F_{p^2} than on short Weierstrass curve over F_p (in the best case up to 28.59% for long multiplication and using complete formula instead of Montgomery ladder). We can also see that for NIST P-256 our method is more efficient than for NIST P-224, because for NIST P-256 we can use irreducible polynomial of form $F(t) = t^2 + 1$, which ensure very fast operations in F_{p^2}.

This solution gives very similar results for these curve as we got in [17] but now we require much less resources.

8 Conclusion

We showed that using F_{p^2} may be very good idea if we want to speed-up hardware implementation of point scalar multiplication on elliptic curves. Firstly we proposed to use twisted Edwards curve over F_{p^2} with fast computable endomorphism using fast F_{p^2} arithmetic. This solution is from 5 to 9 times faster than classic solutions on short Weierstrass curves with similar level of security.

Secondly we showed how we can find for some elliptic curves with cofactor 1 isomorphic twisted Hessian curves in fields extension. For two NIST curves over large prime fields: NIST P-224 and NIST P-256 the degree of such extension is 2 so we can easily use twisted Hessian curve arithmetic over F_{p^2}. Such solution is faster than classic solutions up to 28.5% for NIST P-256 and up to 27.2% for NIST P-224 if we consider solution invulnerable for side channel attacks. We can also use different formula for doubling and addition and then our solution is faster up to 21.4% for NIST P-256 and up to 19.9% for NIST P-224 comparing to classic solutions.

References

[1] NIST. "Recommended Elliptic Curves For Federal Government Use", http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, 1999

[2] R. Gallant, R. Lambert, S. Vanstone. "Faster point multiplication on Elliptic Curves with Efficient Endomorphism". https://www.iacr.org/archive/crypto2001/21390189

[3] H. Edwards. "A normal form for elliptic curves", Bulletin of the American Mathematical Society, Vol. 44, 2007

[4] P. Birkner, M. Joye, T. Lange, Ch. Peters, D. Bernstein. "Twisted Edwards Curves." https://eprint.iacr.org/2008/013

[5] S. Galbraith, X. Lin, M. Scott. "Endomorphisms for Faster Elliptic Curve Cryptography on a Large Class of Curves".https://eprint.iacr.org/2008/194

[6] P. Longa, F. Sica. "Four-Dimensional Gallant-Lambert-Vanstone Scalar Multiplication", https://eprint.iacr.org/2011/608

[7] Z. Hu, P. Longa, M. Xu. "Implementing 4-Dimensional GLV Method on GLS Elliptic Curves with j-Invariant 0". https://eprint.iacr.org/2011/315

[8] S. Ghosh, D. Mukhopadhyay, D. Roychowdhury. "High Speed Flexible Pairing Cryptoprocessor on FPGA Platform", Pairing-Based Cryptography - Pairing 2010, 2010

[9] R. Cheung, S. Duquesne, J. Fan, N. Guillermin, I. Verbauwhede, G. Yao. "FPGA Implementation of Pairings using Residue Number System and Lazy Reduction", Lecture Notes in Computer Sceince, Vol. 6917, 2011

[10] A. Faz-Hernandez, P. Longa, A. Sanchez. "Efficient and Secure Algorithms for GLV-Based Scalar Multiplication and thier implementation on GLV-GLS Curves". https://eprint.iacr.org/2013/158

[11] H. Cohen, G. Frey. "Handbook of Elliptic and Hyperelliptic Curve Cryptography." New York: Chapman & Hall/CRC, 2006.

[12] M. Wang, X. Wang, T. Zhan,. Y. Zheng. "Skew-Frobenius map on twisted Edwards curve". https://eprint.iacr.org/2010/005

[13] T. Lange, D. Bernstein. "Faster addition and doubling on elliptic curves." Lecture Notes in computer Science, Vol. 4833, 2007.

[14] N. Shylashree, V. Sridhar. "Hardware Realization of Fast Multi-Scalar Elliptic Curve Point Multiplication by Reducing the Hamming Weights Over GF(p)", I.J.Computer Network and Information Security, volume 10, 2014

[15] K. Okeya, K. Sakurai. "Fast Multi-scalar Multiplication Methods on Elliptic Curves with Precomputation Strategy Using Montgomery Trick", Lecture Notes in Computer Science, Vol. 2523, 2002

[16] D. Bernstein, Ch. Chuengsatiansup, D. Kohel, T. Lange. "Twisted Hessian curves". https://eprint.iacr.org/2015/781

[17] M. Wroński. "Faster point scalar multiplication on NIST elliptic curves over $GF(p)$ using (twisted) Edwards curves over $GF(p^3)$". http://eprint.iacr.org/2015/977

[18] http://www.iacr.org/archive/pkc2010/60560246/60560246