"Nice" Curves

Kaushik Nath and Palash Sarkar

Applied Statistics Unit
Indian Statistical Institute
203, B. T. Road
Kolkata - 700108
India
{kaushikn_r,palash}@isical.ac.in

Abstract

Within the Transport Layer Security (TLS) Protocol Version 1.3, RFC 7748 specifies elliptic curves targeted at the 128-bit and the 224-bit security levels. For the 128-bit security level, the Montgomery curve Curve25519 and its birationally equivalent twisted Edwards curve Ed25519 are specified; for the 224-bit security level, the Montgomery curve Curve448, the Edwards curve Edwards448 (which is isogenous to Curve448) and another Edwards curve which is birationally equivalent to Curve448 are specified. The contribution of this work is to propose new pairs of Montgomery-Edwards curves at both the 128-bit and the 224-bit security levels. The new curves are *nice* in the sense that they have very small curve coefficients and base points. Compared to the curves in RFC 7748, the new curves lose two bits of security. The main advantage of the new curves over those in RFC 7748 is that for 64-bit implementation, all the reduction steps on the outputs of additions and subtractions in the ladder algorithm can be omitted. For 64-bit implementations on the Skylake and the Kaby Lake processors, about 21% improvement in speed is achieved at the 128-bit security level and about 28% improvement in speed is obtained at the 224-bit security level.

Keywords: Elliptic curve cryptography, Montgomery form, Edwards form, Transport Layer Security.

1 Introduction

Elliptic curves were independently introduced in cryptography by Koblitz [21] and Miller [23]. Since their introduction, a large literature has developed around the theory and application of elliptic curves in cryptography. Presently, elliptic curve cryptography is widely used in practical systems. Several standards and proposals have been put forward by a number of influential organisations [13, 29, 9, 31].

The Transport Layer Security (TLS) Protocol, Version 1.3 [30] has been proposed by the Internet Engineering Task Force. This includes RFC 7748 [22] which specifies certain elliptic curves. The document specifies Montgomery form curves and their birationally equivalent Edwards form curves.

Given a prime p, a parameter $A \in \mathbb{F}_p \setminus \{-2, 2\}$ defines the Montgomery curve $E_{M,A,1} : y^2 = x^3 + Ax^2 + x$. Similarly, a parameter $d \in \mathbb{F}_p \setminus \{0, 1, -1\}$ defines the Edwards curve $E_{E,1,d} : u^2 + v^2 = 1 + du^2v^2$ or the twisted Edwards curve $E_{E,-1,d} : -u^2 + v^2 = 1 + du^2v^2$.

We follow a "power-free" and "subscript-free" naming convention for primes and curves. The prime $2^{251} - 9$ will be denoted as $p251$-9, $2^{255} - 19$ will be denoted as $p255$-19, $2^{444} - 17$ will be denoted as $p444$-17, and $2^{448} - 2^{224} - 1$ will be denoted as $p448$-224-1. A Montgomery curve $E_{M,A,1}$ will be denoted as $M[A]$; an Edwards curve $E_{E,1,d}$ will be denoted as $E[d]$ and a twisted Edwards curve $E_{E,-1,d}$ will be denoted as $\widetilde{E}[d]$. If we wish to emphasize the underlying field $\mathbb{F}_p$, we will write $M[p, A]$, $E[p, d]$ and $\widetilde{E}[p, d]$ instead of $M[A]$, $E[d]$ and $\widetilde{E}[d]$ respectively. In terms of this naming convention, the parameters of the new curves and those in RFC 7748 are shown in Table 1.

Curves proposed in RFC 7748:

Over $p255$-19: The birationally equivalent pair $(M[486662], \widetilde{E}[121665/121666])$ has been proposed. The curve $M[486662]$ is the famous Curve25519 and was introduced in [3]. The curve $\widetilde{E}[121665/121666]$ is the famous Ed25519 curve and was introduced in [5].

Sec Level	Prime	Mont	(h, h_T)	(k, k_T)	Security	Mont Base Pt	Ed	Ed Base Pt
≈ 128	$p251$-9	$M[4698]$	$(4, 4)$	$(\ell - 1, \frac{\ell_T - 1}{2})$	124.5	$(3, \cdot)$	$E[\frac{1175}{1174}]$	$(\cdot, 2)$
	$p255$-19	$M[486662]$	$(8, 4)$	$(\frac{\ell-1}{6}, \frac{\ell_T-1}{2})$	126	$(9, \cdot)$	$\widetilde{E}[\frac{121665}{121666}]$	$(\cdot, 4/5)$
≈ 224	$p444$-17	$M[4058]$	$(4, 4)$	$(\frac{\ell-1}{3}, \ell_T - 1)$	221	$(3, \cdot)$	$E[\frac{1015}{1014}]$	$(\cdot, 2)$
	$p448$-224-1	$M[156326]$	$(4, 4)$	$(\frac{\ell-1}{2}, \frac{\ell_T-1}{4})$	223	$(5, \cdot)$	$E[\frac{39082}{39081}]$	$(\cdot, -3/2)$

Table 1: Parameters of curves. See Section 2.3 for the definition of the parameters.

Over $p448$-224-1: The curves $M[156326]$, $E[39082/39081]$ and $E[-39081]$ have been proposed. The curve $M[156326]$ has been named Curve448 in [22]. The curve $E[-39081]$ was proposed in [17] where it was named Ed448-Goldilocks and in [22], it has been called Edwards448. The isogenies between $M[156326]$ and $E[-39081]$ and the birational equivalence between $M[156326]$ and $E[39082/39081]$ have been identified in [22].

Curve25519 and Ed25519 are targeted at the 128-bit security level while Curve448 and Edwards448 are targeted at the 224-bit security level.

Our Contributions

Through this work we propose two new curves at the 128-bit and the 224-bit security levels.

New curves: We introduce the following pairs of birationally equivalent curves.

Over $p251$-9*:* $(M[4698], E[1175/1174])$.

Over $p444$-17*:* $(M[4058], E[1015/1014])$.

The prime $p251$-9 was considered in [6] where the curve $u^2 + v^2 = 1 - 1174u^2v^2$ was introduced and named Curve1174. The Montgomery curve $(4/1175)y^2 = x^3 + (4/1175 - 2)x^2 + x$ with base point $(4, \cdot)$ was considered as birationally equivalent to Curve1174; the corresponding base point on Curve1174 is $(\cdot, 3/5)$. Using the isogenies given in [11], it can be shown that $M[p251\text{-}9, 4698]$ is 4-isogenous to Curve1174 which was introduced in [6].

To the best of our knowledge, neither $M[p251\text{-}9, 4698]$ nor $E[p251\text{-}9, 1175/1174]$ was earlier considered in the literature. Further, the prime $p444$-17 has not been earlier considered in the literature and so neither $M[p444\text{-}17, 4058]$ nor $E[p444\text{-}17, 1015/1014]$ have been considered in the literature.

Table 1 compares the parameters of the newly proposed curves with those in RFC 7748. Note that the curve coefficients of the new curves are quite small. Also, the fixed base points for the new Montgomery and Edwards curve are also very small. In fact, the fixed base point over both the new Edwards curves is $(\cdot, 2)$. As we explain later, this has a significant effect on the speed of fixed base point scalar multiplication over such curves.

Improvements to Montgomery ladder computation: The improvement is based on working with a slightly smaller prime. Suppose $m = \lceil \log_2 p \rceil$ and elements of $\mathbb{F}_p$ are represented using κ 64-bit words. We show that if $64\kappa - m \geq 3$, then it is possible to omit performing the reduction step on the outputs of all the addition/subtraction operations in the ladder step. This is the major reason for obtaining faster ladder computation modulo $2^{251} - 9$ compared to $2^{255} - 19$ and for obtaining faster ladder computation modulo $2^{444} - 17$ compared to $2^{448} - 2^{224} - 1$.

We put forward the curve $M[p251\text{-}9, 4698]$ as a faster alternative to Curve25519 and the curve $M[p444\text{-}17, 4058]$ as a faster alternative to Curve448. In both cases, the loss in security is about 2 bits. For 64-bit implementations on the Skylake and Kaby Lake processors, the gain in speed of $M[p251\text{-}9, 4698]$ over Curve25519 is about 21% while the gain in speed of $M[p444\text{-}17, 4058]$ over Curve448 is about 28%.

Related Works

In this work, we consider elliptic curves over large prime order fields. We note that elliptic curves over composite order fields have been proposed in the literature [18, 18, 18, 18, 10]. Cryptography over hyperelliptic curves was proposed by Koblitz [18] and there have been concrete proposals for cryptography in

genus 2 [16, 8, 1]. For the same security level, computations over these proposals are faster than over genus one prime order field curves. On the other hand, the security perception for composite order fields and genus two curves is different from that of elliptic curves over prime order fields. It is perhaps due to this perception issue that elliptic curves over prime order fields remain to be of primary interest.

Variable base scalar multiplication over Kummer lines associated with Legendre form elliptic curves have been proposed in the literature [15]. These have very efficient vectorised implementations [20]. So, if applications are targeted primarily for vector implementations, then the curves proposed in [20] will be the primary choice. On the other hand, for non-vectorised implementations, Montgomery curves will be faster.

2 Montgomery and (Twisted) Edwards Form Elliptic Curves

We consider elliptic curves over a field $\mathbb{F}_p$ where p is a prime.

In general, the Montgomery form elliptic curve $E_{M,A,B}$ is given by the equation $E_{M,A,B} : By^2 = x^3 + Ax^2 + x$ with $A \in \mathbb{F}_p \setminus \{-2, 2\}$ and $B \in \mathbb{F}_p \setminus \{0\}$. In general, the twisted Edwards form elliptic curve $E_{E,a,d}$ is given by the equation $E_{E,a,d} : au^2 + v^2 = 1 + du^2v^2$ with $a, d \in \mathbb{F}_p \setminus \{0\}$ and $a \neq d$. If $a = 1$, then the corresponding curve is simply called an Edwards form curve (instead of twisted Edwards form curve). If a is a square and d is not a square in $\mathbb{F}_p$, then the addition formula in $E_{E,a,d}$ is complete [4]. In this case, $E_{E,a,d}$ is called a complete twisted Edwards curve. For further details about Montgomery curves, we refer to [24, 7, 12] and for (twisted) Edwards curves, we refer to [14, 2, 4].

In the following discussion, a full field multiplication (resp. squaring) in $\mathbb{F}_p$ will be denoted as [M] (resp. [S]); if one of the multiplicands is a constant, the resulting multiplication will be denoted as [C].

2.1 Addition on Complete (Twisted) Edwards Curves

Following [19], the extended affine coordinate system is (u, v, t) with $t = uv$. The projective version of this coordinate system is (U, V, T, W) where $u = U/W$, $v = V/W$ and $t = T/W$. Suppose, it is required to add $(U_1 : V_1 : T_1 : W_1)$ and $(U_2 : V_2 : T_2 : W_2)$ to obtain $(U_3 : V_3 : T_3 : W_3)$. The formulas for U_3, V_3, T_3 and W_3 are as follows [19].

$$\left.\begin{array}{rcl} U_3 &=& (U_1V_2 + V_1U_2)(W_1W_2 - dT_1T_2) \\ V_3 &=& (V_1V_2 - aU_1U_2)(W_1W_2 + dT_1T_2) \\ T_3 &=& (U_1V_2 + V_1U_2)(V_1V_2 - aU_1U_2) \\ W_3 &=& (W_1W_2 + dT_1T_2)(W_1W_2 - dT_1T_2). \end{array}\right\} \quad (1)$$

1. Computing V_1V_2 and U_1U_2 and then computing $U_1V_2 + V_1U_2$ as $(U_1 + V_1)(U_2 + V_2) - (U_1U_2 + V_1V_2)$ leads to an algorithm for computing U_3, V_3, T_3 and W_3 using 9[M]+2[C] operations, where the multiplications by the two constants are by a and d. If $a = 1$, then the number of operations is 9[M]+1[C].

2. If $a = -1$, then by first computing $\alpha = (V_1 + U_1)(V_2 + U_2)$, $\beta = (V_1 - U_1)(V_2 - U_2)$ and then computing $2(V_1V_2 + U_1U_2) = \alpha + \beta$ and $2(V_1U_2 + U_1V_2) = \alpha - \beta$, the number of operations can be brought down to 8[M]+1[C] [19], where 1[C] corresponds to a multiplication by d. The relevant formula becomes the following.

$$\left.\begin{array}{rcl} 4U_3 &=& 2(U_1V_2 + V_1U_2)(2W_1W_2 - 2dT_1T_2) = (\alpha - \beta)(2W_1W_2 - 2dT_1T_2) \\ 4V_3 &=& 2(V_1V_2 + U_1U_2)(2W_1W_2 + 2dT_1T_2) = (\alpha + \beta)(2W_1W_2 + 2dT_1T_2) \\ 4T_3 &=& 2(U_1V_2 + V_1U_2)2(V_1V_2 + U_1U_2) = (\alpha - \beta)(\alpha + \beta) \\ 4W_3 &=& (2W_1W_2 + 2dT_1T_2)(2W_1W_2 - 2dT_1T_2). \end{array}\right\} \quad (2)$$

   If $W_1 = 1$, the number of operations required is 7[M]+1[C] [19].

3. For $a = -1$, suppose $(U_1 : V_1 : T_1 : W_1)$ is a fixed base point with $W_1 = 1$. By pre-computing and storing $(V_1 - U_1, V_1 + U_1, 2dT_1)$ the number of operations can be brought down to 7[M] [5]. The multiplication by d becomes part of the pre-computed quantity $2dT_1$. In this formula, since the multiplication by d is part of the pre-computed quantity $2dT_1$, the efficiency of the computation is not affected by whether d is small or large. Also, the efficiency of the computation is not affected by whether V_1 (or U_1) is small or large.

Consider (1) for $a = 1$ and suppose $(U_1 : V_1 : T_1 : W_1)$ is a fixed base point where $W_1 = 1$. Further suppose that V_1 is small and $U_1 + V_1$ and dT_1 are pre-computed and stored as part of $(U_1, V_1, U_1 + V_1, dT_1)$.

In (1), by directly computing U_1V_2, V_1U_2, U_1U_2 and V_1V_2, $(dT_1)T_2$ along with the other four multiplications, the formulas in (1) can be computed using 7[M]+2[C], where 2[C] counts the multiplications V_1U_2 and V_1V_2. The efficiency of the computation following this strategy is not affected by whether d is small or large. For the curves that we introduce, V_1 is equal to 2 as can be seen from Table 1. So, for fixed base multiplication, the difference in the cost between $a = -1$ and $a = 1$ is essentially two multiplications by very small constants.

For dedicated (not unified) addition in $\widetilde{E}[p, d]$, it has been shown in [19] that 8[M] operations are sufficient without the assumption that $(U_1 : V_1 : T_1 : W_1)$ is a fixed base point. The corresponding formulas do not involve d. Further, Section 4.3 of [19] shows how to perform efficient scalar multiplication using fast formulas for dedicated addition and dedicated doubling that do not involve d. The resulting scalar multiplication is not necessarily constant time and can be used only when the scalars are not secret.

Summary:

Role of d: For the fastest formulas, the size of d does not play a role.

- For fixed base point scalar multiplication, the fastest complete addition formulas over both $E[d]$ and $\widetilde{E}[d]$ do not depend on the size of d.
- For scalar multiplication with non-secret scalars, the fastest formulas do not involve d.

Size of fixed base point:

- For $\widetilde{E}[p, d]$, the fastest formula for complete and unified addition does not depend on the size of any of the components of the fixed base point. The number of operations required is 7[M].
- For $E[p, d]$, the fastest formula for complete and unified addition is achieved when V_1 is small. The number of operations required is 7[M]+2[C], where 2[C] counts two multiplications by very small constants. In particular, for both $E[p251\text{-}9, \frac{1175}{1174}]$ and $E[p444\text{-}17, \frac{1015}{1014}]$, $(\cdot, 2)$ is a base point. So, the multiplication by constant is the operation of multiplying an element of $\mathbb{F}_p$ by 2.

2.2 Birational Equivalences between Montgomery and Edwards Curves

Consider the curves $M[A]$ and $E[d]$ over a field $\mathbb{F}_p$ with $p \equiv 3 \bmod 4$. If $A - 2$ is a square in $\mathbb{F}_p$, then the map

$$(x, y) \quad \mapsto (u, v) \quad = (\delta x/y, (x + 1)/(x - 1)), \tag{3}$$

where $\delta^2 = (A - 2)$, is a birational equivalence from $M[A]$ to $E[d]$ with exceptional points $y = 0$ and $x = 1$. Conversely, the map

$$(u, v) \mapsto (x, y) = ((v + 1)/(v - 1), \delta(v + 1)/(u(v - 1))), \tag{4}$$

is a birational equivalence from $E[d]$ to $M[A]$ with exceptional points $u = 0$ and $v = 1$. The relation between A and d is $(A - 2)/4 = 1/(d - 1)$. The above birational equivalences can be obtained using the elementary birational equivalences in [2, 4]. On the other hand, verification of these birational equivalences can be done by direct substitution.

2.3 Security Properties

Let E be an elliptic curve over $\mathbb{F}_p$, where p is a prime.

Let $n = \#E(\mathbb{F}_p)$ and $n_T = 2(p+1) - \#E(\mathbb{F}_p)$, i.e., n and n_T are the orders of $E(\mathbb{F}_p)$ and its twist. Let ℓ (resp. ℓ_T) be a prime such that $n = h \cdot \ell$ (resp. $n_T = h_T \cdot \ell_T$). Cryptography is done over a subgroup of $E(\mathbb{F}_p)$ of size ℓ. The parameters h and h_T are the co-factors of $E(\mathbb{F}_p)$ and its twist respectively.

For a Montgomery curve, the curve order n is a multiple of 4. Using this fact along with $n + n_T = 2(p + 1)$, it is easy to argue that if $p \equiv 3 \bmod 4$, then the minimum value of (h, h_T) is $(4, 4)$, while if $p \equiv 1 \bmod 4$, then the minimum value of (h, h_T) is either $(8, 4)$ or $(4, 8)$.

Let k (resp. k_T) be the smallest positive integer such that $\ell | p^k - 1$ (resp. $\ell_T | p^{k_T} - 1$). The parameters k and k_T are the embedding degrees of the curve and its twist respectively.

The complex multiplication field discriminant D of E is defined in the following manner. Let $t = p+1-n$. By Hasse's theorem, $|t| \leq 2\sqrt{p}$ and in the cases that we considered $|t| < 2\sqrt{p}$ so that $t^2 - 4p$ is a

negative integer; let s^2 be the largest square dividing t^2-4p; define $D=(t^2-4p)/s^2$ if $t^2-4p \bmod 4 = 1$ and $D=4(t^2-4p)/s^2$ otherwise.

SafeCurves[1], recommend all of ℓ, ℓ_T, k, k_T and D to be large. In particular, we are interested in curves for which (h, h_T) has the optimal value.

By security of a curve in terms of bits we will mean the value of the expression $\frac{1}{2}\min(\log_2 \ell, \log_2 \ell_T)$.

3 Concrete Curves

The parameters of the new curves are given below.

Curves over $\mathbb{F}_{2^{251}-9}$: Let $p = 2^{251}-9 \equiv 3 \bmod 4$. The minimum positive value of A for which the curve $M[p251\text{-}9, A]$ attains the optimal value of (h, h_T) is $A = 4698$. We have that $A-2$ is a square in $\mathbb{F}_p$. Using the birational equivalences given by (3) and (4), we obtain the pair $(M[4698], E[1175/1174])$ of birationally equivalent curves.

The parameters for $M[p251\text{-}9, 4698]$ are as follows.

$$
\begin{aligned}
n &= 36185027886661311069865932815214971203693561411179818960939570470945719024 04,\\
\ell &= 90462569716653277674664832038037428009233903527949547402348926177364297560 1,\\
\log_2 \ell &= 249,\\
h &= 4,\\
k &= \ell - 1,\\
n_T &= 36185027886661311069865932815214971204600179004845533563721419533999987000 76,\\
\ell_T &= 90462569716653277674664832038037428011500447512113833909030548834999967501 9,\\
\log_2 \ell_T &= 249,\\
h_T &= 4,\\
k_T &= (\ell_T - 1)/2,\\
D &= -12419122501803997450343277787015672473799971462290478421477646400945935050060,\\
\lceil \log_2(-D) \rceil &= 253.
\end{aligned}
$$

The point $(\cdot, 2)$ is a point of order ℓ on $E_{E,1,1175/1174}$; the corresponding point on $E_{M,4698,1}$ is $(3, \cdot)$.

The set of scalars for $E_{M,4698,1}$ is set to be $4(2^{248} + \{0, 1, \ldots, 2^{248}-1\})$. Given a 32-byte scalar a, the clamping function $\mathsf{clamp}(a)$ is defined as follows (assuming that the first byte is the least significant byte of a): clear bits 0 and 1 of the first byte; set bit number 2 of the last byte and clear bits numbered 3 to 7 of the last byte.

Curves over $\mathbb{F}_{2^{444}-17}$: Let $p = 2^{444}-17 \equiv 3 \bmod 4$. The minimum positive value of A for which the curve $M[p444\text{-}17, A]$ attains the optimal value of (h, h_T) is $A = 4058$. We have that $A-2$ is a square in $\mathbb{F}_p$. Using the birational equivalences given by (3) and (4), we obtain the pair $(M[4058], E[1015/1014])$

[1] https://safecurves.cr.yp.to/disc.html, accessed on September 8, 2019.

of birationally equivalent curves. The parameters for $M[p444\text{-}17, 4058]$ are as follows.

$$
\begin{aligned}
n &= 4542742026847543065933273799300028339710258504295737876759313744878847\backslash \\
&\quad 8221099948877847233254577748571252041451263610502018101866494 52, \\
\ell &= 1135685506711885766483318449825007084927564626073934469189828436\backslash \\
&\quad 1971197055274987219461808313644437142813010362815902625504525466623 63, \\
\log_2 \ell &= 442, \\
h &= 4, \\
k &= (\ell - 1)/3, \\
n_T &= 4542742026847543065933273799300028339710258504295737876759313744879\backslash \\
&\quad 1432192064745527989158013762670837970111055656911191490015016927348, \\
\ell_T &= 1135685506711885766483318449825007084927564626073934469189828436\backslash \\
&\quad 2197858048016186381997289503440667709492527763914227797872503754231837, \\
\log_2 \ell_T &= 442, \\
h_T &= 4, \\
k_T &= (\ell_T - 1), \\
D &= -1795290825514957729951691737953552054640775333171791787428768650737 4\backslash \\
&\quad 0299495896775961634341986909858530888358546664503937673912260606892, \\
\lceil \log_2(-D) \rceil &= 446.
\end{aligned}
$$

The point $(\cdot, 2)$ is a point of order ℓ for $E_{E,1,1015/1014}$; the corresponding point on $E_{M,4058,1}$ is $(3, \cdot)$.

The set of scalars is set to be $4(2^{441} + \{0, 1, \ldots, 2^{441} - 1\})$. Given a 56-byte scalar a, the clamping function $\mathsf{clamp}(a)$ is defined as follows (assuming that the first byte is the least significant byte of a): clear bits 0 and 1 of the first byte; set bit number 3 of the last byte and clear bits numbered 4 to 7 of the last byte.

Remark: Using the isogenies given in [11], it can be shown that $M[p444\text{-}17, 4058]$ is 4-isogenous to $E[p444\text{-}17, -1014]$. Also, it has been mentioned earlier that $M[p251\text{-}9, 4698]$ is 4-isogenous to Curve1174. Connecting Montgomery and Edwards using these isogenies can be a problem, since a small base point on one of these curves does not translate to a small base point on the other.

4 Implementation

In this section, we discuss about the implementation issues.

4.1 Representation of Field Elements

Let $m = \lceil \log_2 p \rceil$. Elements of $\mathbb{F}_p$ can be represented as m-bit strings. We are interested in 64-bit multi-precision arithmetic. Elements of $\mathbb{F}_p$ are considered to be κ 64-bit words. Conventionally, each such word is called a limb. We will consider packed or saturated limb representation. In this representation, m is written as $m = \eta(\kappa - 1) + \nu$ with $1 \leq \nu \leq \eta$, where $\eta = 64$. In other words, the first $\kappa - 1$ limbs are 64 bits long while the last limb is between 1 and 64 bits long.

The representations of the four primes of interest to this work are given in Table 2. Note that for $p251\text{-}9$ and $p444\text{-}17$, $64\kappa - m \geq 3$ (equivalently, the last limb has three or more "free" bits), for $p255\text{-}19$, $64\kappa - m = 1$ (equivalently, the last limb has one "free" bits) and for $p448\text{-}224\text{-}1$, $64\kappa = m$ (equivalently, the last limb has no "free" bits). These have significant effect on the ladder computation as we will see below.

Prime	m	κ	η	ν	$64\kappa - m$
$p251\text{-}9$	251	4	64	59	5
$p255\text{-}19$	255	4	64	63	1
$p444\text{-}17$	444	7	64	60	4
$p448\text{-}224\text{-}1$	448	7	64	64	0

Table 2: Saturated limb representations of primes related to this work.

Remark: For 64-bit arithmetic, it is possible to work with representations where $\eta < 64$. For example, a 5-limb representation with each limb having 51 bits has been proposed for $2^{255}-19$. Such representations can be considered to have redundant or unsaturated limb representation. The target architecture for our implementation is Skylake and later processors. For these architectures, implementations using the saturated limb representation outperform the implementations using unsaturated limb representation. So, we do not consider unsaturated limb representations in this work.

4.2 Integer Multiplication/Squaring

The Skylake and later processors provide the `mulx/adcx/adox` instructions. The `mulx` instruction performs a multiplication without affecting the carry and overflow flags. The `adcx` instruction adds with carry using the carry flag but, does not affect the overflow flag, while the `adox` instructions adds with carry using the overflow flag but, does not affect the carry flag. We collectively term the set `mulx/adcx/adox` to be the `maax` instructions. The availability of `maax` operations opened up the possibility of very fast integer multiplication using two double carry chains. For multiplication/squaring of 256-bit numbers, this has been explained in the Intel white papers [28, 27]. A general algorithmic description for multiplication/squaring of 64κ-bit numbers, $\kappa \geq 4$ is given in [25].

4.3 Reduction

Integer multiplication/squaring of κ-limb quantities produces a 2κ-limb output. The reduction step reduces this output modulo the prime p. A full reduction will reduce the output to a value less than p. For the purposes of efficiency a full reduction is not carried out in the intermediate steps of the computation. Instead a size reduction is done. The size reduction can be of two types, namely, reduction to an $(m+1)$-bit integer and reduction to an m-bit integer (note than an m-bit integer is not necessarily fully reduced since it is not necessarily less than p). The former is more efficient than the later. Further, the reduced quantity should again be a κ-limb quantity. If $\nu < 64$, i.e., the last limb has at least one free bit, then reduction to an $(m+1)$-bit integer is a κ-limb quantity. On the other hand, if $\nu = 64$, i.e., the last limb has no free bits, then it is a necessity to reduce to an m-bit integer to obtain a κ-limb quantity. Among the primes in Table 2, the prime $2^{448}-2^{224}-1$ has no extra bits in the last limb and the reduction for this prime has to be to an m-bit integer. For the other primes, it is possible to reduce to an $(m+1)$-bit integer without any overfull. The size reductions to $(m+1)$ bits modulo $2^{251}-9$ and $2^{444}-17$ have been done following the algorithm reduceSLPMP in [25].

4.4 Addition and Subtraction

Other than multiplication/squarings, the ladder algorithm also uses field addition and subtraction. In the ladder algorithm, the inputs to an addition/subtraction operation are outputs of multiplication/squaring operations and the outputs of addition/subtraction operations are inputs to multiplication/squaring operations. In particular, the outputs of addition/subtraction are never inputs to another addition/subtraction.

We have mentioned that the outputs of multiplication/squaring are size reduced to either m bits or to $(m+1)$ bits. So, the inputs to addition/subtraction operations are either m bits or $(m+1)$ bits. We require the outputs of the addition/subtraction operations to be κ-limb quantities so that the integer multiplication/squaring algorithm can be applied to these outputs. So, it is not always required to size reduce the outputs of addition/subtraction operations to m or $(m+1)$ bits. Depending upon the sizes of the inputs to the addition/subtraction operation and the relative values of η and ν, various cases may arise. We discuss the cases of addition and subtraction separately.

Addition: A field addition is typically an integer addition followed by a possible reduction operation. The integer addition operation increases the size of the output by one bit compared to the sizes of the inputs.

Case p448-224-1: In this case, there is no leeway in the last limb and the output of integer addition must necessarily be reduced to obtain a κ-limb quantity.

Case p255-19: If the inputs to the addition are m-bit quantities, then it is possible to omit applying the reduction step to the output of the integer addition operation. On the other hand, if the inputs to the addition are $(m+1)$-bit quantities, then the reduction step has to be applied to the output of the integer addition operation. The inputs to the addition operation are the outputs of previous multiplication/squaring operations. So, whether the output of the integer addition needs

to be reduced depends on whether the outputs of the multiplication/squaring operation have been reduced to m bits or to $(m+1)$ bits.

Cases $p251$-9 *and* $p444$-17*:* In these cases, it is possible to reduce the outputs of multiplication/squaring to $(m+1)$ bits *and* omit the reduction step after the integer addition operation.

Subtraction: A field subtraction is of the type $a - b \bmod p$. To avoid handling negative numbers, a suitable multiple of p is added to a so that the result is guaranteed to be positive. Since the result will be reduced modulo p, the correctness of the result is not affected by adding a multiple of p.

Cases $p255$-19 *and* $p448$-224-1*:* The reduction operation must be performed on the output of each subtraction operation to ensure that the result fits in κ limbs.

Cases $p251$-9 *and* $p444$-17*:* The operation $a - b \bmod p$ is performed as follows. Note that both a and b are $(m+1)$-bit quantities. The operation $4p + a - b$ is guaranteed to be an $(m+3)$-bit non-negative integer. So, instead of performing $a - b \bmod p$, the operation $(4p + a) - b$ is computed. Since the result is at most an $(m+3)$-bit quantity, it fits within κ limbs. Consequently, no reduction operation is performed on this result.

Remark: We have discussed the issue of avoiding reduction with respect to 64-bit arithmetic. The general idea, on the other hand, holds for saturated limb representations using 32-bit (or, lower) arithmetic. The implementation benefits of of $p251$-9 and $p444$-17 over $p255$-19 and $p448$-224-1 also holds for 32-bit arithmetic.

4.5 Optimisations of the Ladder Step

Based on the description in Section 4, the following strategy may be adopted for implementing the ladder step for the various primes.

Case $p448$-224-1*:* The outputs of all multiplication/squaring operations are to be size reduced to m bits. Outputs of all addition/subtraction operations are to be size reduced to m bits.

Case $p255$-19*:* The outputs of all multiplication/squaring operations are to be size reduced to $(m+1)$ bits. Outputs of all addition/subtraction operations are to be size reduced to m or $(m+1)$ bits.

Cases $p251$-9 *and* $p444$-17*:* The outputs of all multiplication/squaring operations are to be size reduced to $(m+1)$ bits. Outputs of all addition/subtraction operations are left unreduced.

The above strategy has direct consequences to the efficiencies of the ladder step for the various primes. We summarise these below.

4-*limb representations:* For both $\mathbb{F}_{2^{251}-9}$ and $\mathbb{F}_{2^{255}-19}$, field elements have 4-limb representations. So, the integer multiplication/squaring operations take the same time in both cases. Due to the ability to avoid reductions, the ladder step is significantly faster modulo $2^{251} - 9$ compared to $2^{255} - 19$.

7-*limb representations:* For the fields $\mathbb{F}_{2^{444}-17}$ and $\mathbb{F}_{2^{448}-2^{224}-1}$, field elements have 7-limb representations. So, the integer multiplication/squaring operations take the same time in both cases. Due to the ability to avoid reductions, the ladder step is significantly faster modulo $2^{444} - 17$ compared to $2^{448} - 2^{224} - 1$.

The Intel x86 64-bit assembly codes implementing the Montgomery ladder for the proposed curves are publicly available at the following link:

https://github.com/kn-cs/nice-curves

4.6 Timings

The timing experiments were carried out on a single core of Skylake and Kaby Lake processors. During measurement of the cpu-cycles, turbo-boost and hyper-threading features were turned off. An initial cache warming was done with 25000 iterations and then the median of 100000 iterations was recorded. The time stamp counter TSC was read from the CPU to RAX and RDX registers by RDTSC instruction.

Curve	Field	Security	Skylake	Kaby Lake	Reference
Curve25519	$\mathbb{F}_{2^{255}-19}$	126	118231	113728	[26]
Curve448	$\mathbb{F}_{2^{448}-2^{224}-1}$	223	536362	521934	[26]
$M[p251\text{-}9, 4698]$	$\mathbb{F}_{2^{251}-9}$	124.5	92250	88882	This work
$M[p444\text{-}17, 4058]$	$\mathbb{F}_{2^{444}-17}$	221	384905	365382	This work

Table 3: CPU-cycle counts on Skylake and Kaby Lake processors for variable base scalar multiplication on the Montgomery form curves.

Platform specifications: The details of the hardware and software tools used in our software implementations are as follows.

Skylake: Intel® Core™ i7-6500U 2-core CPU @ 2.50GHz. The OS was 64-bit Ubuntu 14.04 LTS and the source code was compiled using GCC version 7.3.0.

Kaby Lake: Intel® Core™ i7-7700U 4-core CPU @ 3.60GHz. The OS was 64-bit Ubuntu 18.04 LTS and the source code was compiled using GCC version 7.3.0.

Timings in form of cpu-cycles are shown in Table 3 for Skylake and Kaby Lake processors. For comparison, we provide the timings of the most efficient (to the best of our knowledge) publicly available 64-bit implementations. Such implementations consist of the code corresponding to the work [26]. The timings of the previous implementations were obtained by downloading the relevant software and measuring the required cycles on the same platforms where the present implementations have been measured. Based on Table 3, we observe that $M[p251\text{-}9, 4698]$ is about 21% to 22% faster than Curve25519, while $M[p444\text{-}17, 4058]$ is about 28% to 30% faster than Curve448.

5 Conclusion

In this paper, we have introduced two pairs of Montgomery-Edwards curves for performing cryptography at the 128-bit and the 224-bit security levels. Compared to the curves proposed in IETF RFC 7748, the new curves provide 2 bits less security. The advantage is that for 64-bit implementations on Skylake and Kaby Lake processors, the gain in speed is about 21% at the 128-bit security level and about 28% at the 224-bit security level.

Acknowledgements: Thanks to Rene Struik for comments on the paper and to Armando Faz Hernández for comments on our implementation code.

References

[1] D. J. Bernstein, C. Chuengsatiansup, T. Lange, and P. Schwabe. Kummer strikes back: New DH speed records. In *Advances in Cryptology - ASIACRYPT*, volume 8873 of *Lecture Notes in Computer Science*, pages 317–337. Springer, 2014.

[2] D. J. Bernstein and Lange T. Faster addition and doubling on elliptic curves. In *Advances in Cryptology - ASIACRYPT*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.

[3] Daniel J. Bernstein. Curve25519: New Diffie-Hellman Speed Records. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[4] Daniel J. Bernstein, Peter Birkner, Marc Joye, Tanja Lange, and Christiane Peters. Twisted edwards curves. In Serge Vaudenay, editor, *Progress in Cryptology - AFRICACRYPT 2008, First International Conference on Cryptology in Africa, Casablanca, Morocco, June 11-14, 2008. Proceedings*, volume 5023 of *Lecture Notes in Computer Science*, pages 389–405. Springer, 2008.

[5] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *J. Cryptographic Engineering*, 2(2):77–89, 2012.

[6] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 967–980. ACM, 2013.

[7] Daniel J. Bernstein and Tanja Lange. Montgomery curves and the Montgomery ladder. In Joppe W. Bos and Arjen K. Lenstra, editors, *Topics in Computational Number Theory inspired by Peter L. Montgomery*, pages 82–115. Cambridge University Press, 2017.

[8] Joppe W. Bos, Craig Costello, Hüseyin Hisil, and Kristin E. Lauter. Fast cryptography in genus 2. In *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 194–210. Springer, 2013.

[9] Brainpool. ECC standard. http://www.ecc-brainpool.org/ecc-standard.htm.

[10] C. Costello and P. Longa. Four($\mathbb{Q}$): Four-dimensional decompositions on a $\mathbb{Q}$-curve over the Mersenne prime. In *Advances in Cryptology - ASIACRYPT Part I*, volume 9452 of *Lecture Notes in Computer Science*, pages 214–235. Springer, 2015.

[11] Craig Costello and Michael Naehrig. Isogenies between (twisted) Edwards and Montgomery curves. https://cryptosith.org/papers/isogenies_tEd2Mont.pdf, 2015. Accessed on 16 September, 2019.

[12] Craig Costello and Benjamin Smith. Montgomery curves and their arithmetic - the case of large characteristic fields. *J. Cryptographic Engineering*, 8(3):227–240, 2018.

[13] NIST Curves. Recommended elliptic curves for federal government use. http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf, 1999.

[14] Harold M. Edwards. A Normal Form for Elliptic Curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.

[15] P. Gaudry and D. Lubicz. The arithmetic of characteristic 2 Kummer surfaces and of elliptic Kummer lines. *Finite Fields and Their Applications*, 15(2):246–260, 2009.

[16] P. Gaudry and É. Schost. Genus 2 point counting over prime fields. *J. Symb. Comput.*, 47(4):368–400, 2012.

[17] Mike Hamburg. Ed448-goldilocks, a new elliptic curve. *IACR Cryptology ePrint Archive*, 2015:625, 2015.

[18] Darrel Hankerson, Koray Karabina, and Alfred Menezes. Analyzing the Galbraith-Lin-Scott point multiplication method for elliptic curves over binary fields. *IEEE Trans. Computers*, 58(10):1411–1420, 2009.

[19] Hüseyin Hisil, Kenneth Koon-Ho Wong, Gary Carter, and Ed Dawson. Twisted Edwards curves revisited. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, volume 5350 of *Lecture Notes in Computer Science*, pages 326–343. Springer, 2008.

[20] Sabyasachi Karati and Palash Sarkar. Kummer for Genus One over Prime Order Fields. *Journal of Cryptology*, 2019. https://doi.org/10.1007/s00145-019-09320-4.

[21] Neal Koblitz. Elliptic curve cryptosystems. *Math. Comp.*, 48(177):203–209, 1987.

[22] Adam Langley and Mike Hamburg. Elliptic curves for security. Internet Research Task Force (IRTF), Request for Comments: 7748, https://tools.ietf.org/html/rfc7748, 2016. Accessed on 16 September, 2019.

[23] Victor S. Miller. Use of elliptic curves in cryptography. In *Advances in Cryptology - CRYPTO'85, Santa Barbara, California, USA, August 18-22, 1985, Proceedings*, pages 417–426. Springer Berlin Heidelberg, 1985.

[24] Peter L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48(177):243–264, 1987.

[25] Kaushik Nath and Palash Sarkar. Efficient Arithmetic in (Pseudo-)Mersenne Prime Order Fields. *IACR Cryptology ePrint Archive*, 2018:985, 2018.

[26] Thomaz Oliveira, Julio López Hernandez, Hüseyin Hisil, Armando Faz-Hernández, and Francisco Rodríguez-Henríquez. How to (pre-)compute a ladder - improving the performance of X25519 and X448. In Carlisle Adams and Jan Camenisch, editors, *Selected Areas in Cryptography - SAC 2017 - 24th International Conference, Ottawa, ON, Canada, August 16-18, 2017, Revised Selected Papers*, volume 10719 of *Lecture Notes in Computer Science*, pages 172–191. Springer, 2017.

[27] E. Ozturk, J. Guilford, and V. Gopal. Large integer squaring on Intel architecture processors, intel white paper. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/large-integer-squaring-ia-paper.pdf, 2013.

[28] E. Ozturk, J. Guilford, V. Gopal, and W. Feghali. New instructions supporting large integer arithmetic on Intel architecture processors, intel white paper. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/ia-large-integer-arithmetic-paper.pdf, 2012.

[29] Certicom Research. SEC 2: Recommended elliptic curve domain parameters. http://www.secg.org/sec2-v2.pdf, 2010.

[30] Version 1.3 TLS Protocol. RFC 8446. https://datatracker.ietf.org/doc/rfc8446/?include_text=1, 2018. Accessed on 16 September, 2019.

[31] NUMS: Nothing up my sleeve. https://tools.ietf.org/html/draft-black-tls-numscurves-00.