A note on high-security general-purpose elliptic curves

Diego F. Aranha[1], Paulo S. L. M. Barreto[2⋆], and Geovandro C. C. F. Pereira[2]

[1] Computer Science Dept, University of Brasília.
E-mail: dfaranha@unb.br
[2] Escola Politécnica, University of São Paulo.
E-mails: {pbarreto,geovandro}@larc.usp.br

Abstract. In this note we describe some general-purpose, high-efficiency elliptic curves tailored for security levels beyond 2^{128}. For completeness, we also include legacy-level curves at standard security levels. The choice of curves was made to facilitate state-of-the-art implementation techniques.

1 Introduction

General-purpose elliptic curves are necessary to attain high-efficiency implementations of the most common cryptographic protocols like asymmetric encryption and plain digital signatures (but setting aside less conventional application like identity-based encryption). The standard NIST curves [13], though fairly efficient overall, arguably no longer represent the state of the art in the area [4, 6].

More efficient general-purpose curves have been recently proposed to address this situation [3, 4, 7], but for the 2^{128} security level at most, which corresponds to the expected security level of the standard NIST curve P-256 (or its binary counterpart, B-283). This is the case of Curve25519 [3] and Curve1174 [4]. However, while there is reason to look for higher security curves [14], no similar curves seem to have been proposed in the literature for higher security levels, matching the presumed levels of (say) the standard NIST curves P-384 and/or P-521.

In this short note we address this need up to the expected security level of P-384, adopting the same settings as Curve25519 and Curve1174, respectively.

2 Curve choice

The curves Curve25519 and Curve1174 have been engineered to facilitate simple, efficient and secure implementation of general-purpose elliptic curve cryptosystems, with impressive results [7]. On these grounds, it makes sense to look for similar curves at higher security levels. At the same time, one can take the opportunity to provide curves matching the expected security levels of other standard

⋆ Supported by CNPq research productivity grant 306935/2012-0.

NIST curves, the most prominent case being P-224 (and its binary counterpart, B-233), corresponding approximately to the security level of 3DES [11].

Curve25519 [3] is an Elligator type 2 curve with the following properties (among others):

- It is a Montgomery curve [12] over a large prime field $\mathbb{F}_p$;
- The prime p has the form $p = 2^m - \delta$ where $0 < \delta < \lceil \lg(p) \rceil = m$;
- The prime p satisfies $p \equiv 5 \pmod 8$, hence square root computation in $\mathbb{F}_p$ can be done with the Atkin method [2];
- The value $\xi = 2$ is a quadratic non-residue in $\mathbb{F}_p$, and hence can used to define a non-trivial quadratic twist of an elliptic curve over $\mathbb{F}_p$;
- The curve equation is $E : y^2 = x^3 + Ax^2 + x$ and the twist equation is $E' : v^2 = u^3 + 2Au^2 + 4u$, where $A > 2$ is as small as possible.
- The curve order has the form $n = 8r$ where r is prime;
- The order of the non-trivial quadratic twist of the curve has the form $n' = 4r'$ where r' is prime, with $|r'| = |r| + 1$;

Curve1174 [4] is an Elligator type 1 curve with the following properties (among others):

- It is an Edwards curve [5,9] over a large prime field $\mathbb{F}_p$;
- The prime p has the form $p = 2^m - \delta$ where $0 < \delta < \lceil \lg(p) \rceil = m$;
- The prime p satisfies $p \equiv 3 \pmod 4$, hence square root computation in $\mathbb{F}_p$ can be done with the Cippolla-Lehmer method [10];
- The curve equation is $E : x^2 + y^2 = 1 + dx^2y^2$ and the equation of a non-trivial quadratic twist of E is $E' : u^2 + v^2 = 1 + (1/d)u^2v^2$, where $d > 1$ is as small as possible;
- The curve order has the form $n = 4r$ where r is prime;
- The order of the non-trivial quadratic twist of the curve has the form $n' = 4r'$ where r' is prime, with $|r'| = |r|$;

3 The curves

We now list curves for several security levels, up to the level roughly comparable to the presumed security level of the NIST curve P-384. The primes have the general form $p = 2^m - \delta$ for δ as small as possible. While it would be desirable that $\delta < 32$ (see [4]), this is not always possible. Yet, insisting that $\delta < \lg p$ increases the likeliness that any attack advantage this setting might cause is negligible (exponentially small). An additional practical constraint is that the value of δ fits one byte, to facilitate the detection of values outside the valid range is this is deemed necessary.

Table 1 contains Montgomery curves, while Table 2 contains Edwards curves. For completeness, we include the original Curve25519 and Curve1174. The prime group order is r.

A proof-of-concept implementation of all these curves is available as part of the RELIC library [1]. Work on a production-quality implementation is ongoing.

Table 1. Montgomery curves

curve	p	A	$\|r\|$	security	r
Curve22103	$2^{221} - 3$	204400	218	$2^{108.8}$	42124916667422874679167211073468\ 15224122133594969171679448734187\ 89
Curve25519	$2^{255} - 19$	486662	252	$2^{125.8}$	72370055773322622139731865630429\ 94240857116359379907606001950938\ 285454250989
Curve383187	$2^{383} - 187$	229969	380	$2^{189.8}$	24626253872746549507674400062589\ 75862817483704404090416747124418\ 61257488060594435036992487765060\ 6926799392131911201

Table 2. Edwards curves

curve	p	$-d$	$\|r\|$	security	r
Curve4417	$2^{226} - 5$	4417	224	$2^{111.8}$	26959946667150639794667015087019\ 63059840670773950791945064254090\ 7951
Curve1174	$2^{251} - 9$	1174	249	$2^{124.3}$	90462569716653277674664832038037\ 42800923390352794954740234892617\ 73642975601
Curve67254	$2^{382} - 105$	67254	380	$2^{189.8}$	24626253872746549507674400062589\ 75862817483704404090416745738034\ 55766305456464917126265932668324\ 4604346084081047321

4 Conclusion

We have described general-purpose high-efficiency curves roughly matching the expected security of the standard NIST curves P-384, and as a bonus, also curves roughly matching the expected security of the standard NIST curves P-224. All curves follow the Elligator (1 and 2) strategy, which is arguably the state of the art for the design of cryptographically-oriented elliptic curves.

This is work in progress. Better curves may be suggested as they become available.

References

1. D. F. Aranha and C. P. L. Gouvêa. RELIC is an Efficient LIbrary for Cryptography.
2. O. Atkin. Square roots and cognate matters modulo $p = 8n + 5$. Number Theory mailing list, 1992. http://listserv.nodak.edu/scripts/wa.exe?A2=ind9211&L=nmbrthry&O=T&P=562.
3. Dan J. Bernstein. Curve25519: New Diffie-Hellman speed records. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography – PKC 2006*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.
4. Dan J. Bernstein, M. Hamburg, A. Krasnova, and T Lange. Elligator: Elliptic-curve points indistinguishable from uniform random strings. IACR Cryptology ePrint Archive, report 2013/325, 2013.
5. Dan J. Bernstein and Tanja Lange. Security dangers of the NIST curves. In K. Kurosawa, editor, *Advances in Cryptology – Asiacrypt 2007*, volume 4833 of *Lecture Notes in Computer Science*, pages 29–50. Springer, 2007.
6. Dan J. Bernstein and Tanja Lange. Security dangers of the NIST curves. Invited talk, International State of the Art Cryptography Workshop, Athens, Greece, 2013.
7. Dan J. Bernstein, Tanja Lange, and Peter Schwabe. The security impact of a new cryptographic library. In Alejandro Hevia and Gregory Neven, editors, *Latincrypt 2012*, volume 7533 of *Lecture Notes in Computer Science*, pages 159–176. Springer, 2012.
8. Wieb Bosma, John Cannon, and Catherine Playoust. The Magma algebra system. I. The user language. *J. Symbolic Comput.*, 24(3-4):235–265, 1997. Computational algebra and number theory (London, 1993).
9. Harold M. Edwards. A normal form for elliptic curves. *Bulletin of the American Mathematical Society*, 44:393–422, 2007.
10. D. H. Lehmer. Computer technology applied to the theory of numbers. In W. J. LeVeque, editor, *Studies in Number Theory*. Mathematical Association of America, 1969.
11. Stefan Lucks. Attacking triple encryption. In Serge Vaudenay, editor, *Fast Software Encryption – FSE 1998*, volume 1372 of *Lecture Notes in Computer Science*, pages 239–253. Springer, 1998.
12. P. L. Montgomery. Speeding the Pollard and elliptic curve methods of factorization. *Mathematics of Computation*, 48:243–264, 1987.
13. National Institute of Standards and Technology – NIST. *Federal Information Processing Standard (FIPS 186-4) – Digital Signature Standard (DSS)*, July 2013.
14. National Security Agency – NSA. *Suite B Cryptography / Cryptographic Interoperability*, January 2009. http://www.nsa.gov/ia/programs/suiteb_cryptography/index.shtml.

A Legacy-level curves

Under certain circumstances where one needs a lower but still reasonable security level, or must adhere to layout constraints of legacy applications, or simply cannot afford higher-security curves for lack of computational resources (as may be the case on certain very constrained platforms typical of the Internet of Things), curves matching the expected security level of (say) the NIST curves B-163 or P-192 may be useful.

Though the primary purpose of this note is to suggest some curves at high security levels, for completeness we list a few possible alternatives for those legacy-level curves on Tables 3 and 4.

Table 3. Legacy-level Montgomery curves

curve	p	A	$\|r\|$	security	r
Curve15991	$2^{159} - 91$	82495	156	$2^{77.8}$	91343852333181432387730243585855\ 885116525414589
Curve19119	$2^{191} - 19$	30327	188	$2^{93.8}$	39231885846166754773973683895852\ 5248322907271238795811731

Table 4. Legacy-level Edwards curves

curve	p	A	$\|r\|$	security	r
Curve42000	$2^{157} - 133$	42000	155	$2^{77.3}$	45671926166590716193865246478592\ 509883108923719
Curve65896	$2^{191} - 69$	65896	189	$2^{94.3}$	78463771692333509547947367790096\ 2674043917376225288258601

B Verifying the curves

The following Magma [8] script checks that the curves presented in this note do indeed satisfy the requirements in Section 2, except the condition that the coefficients A and d in the curve equations $E : y^2 = x^3+Ax^2+x$ and $E : x^2+y^2 = 1+dx^2y^2$ are as small as possible in absolute integer value. Extending the script so as to check this last condition is straightforward, but the processing time can be very long (several weeks for the highest security levels, if run sequentially). Independent verification has been kindly provided in Sage by S. Neves at `http://eden.dei.uc.pt/~sneves/647.sage`.

```
function MontyCurve(m, A)
    p := 2^m;
```

```
repeat
    p := PreviousPrime(p);
until p mod 8 eq 5;
delta := 2^m - p;
if delta gt m then
    return false;
end if;
F := GF(p);
z := 2;
if IsSquare(F!z) then
    return false;
end if;
d := (F!A - 2)/(F!A + 2);
if IsSquare(F!A - 2) or IsSquare(F!A^4 - 4) or IsSquare(d) then
    return false;
end if;
// check curve y^2 = x^3 + A*x^2 + x:
ok, E := IsEllipticCurve([0, F!A, 0, 1, 0]);
if not ok then
    return false;
end if;
n := #E;
if (n mod 8 ne 0) or not IsProbablePrime(n div 8) then
    return false;
end if;
// check twist v^2 = u^3 + A*z*u^2 + z^2*u:
ok, Et := IsEllipticCurve([0, F!A*z, 0, F!z^2, 0]);
if not ok then
    return false;
end if;
nt := #Et;
if (nt mod 4 ne 0) or not IsProbablePrime(nt div 4) then
    return false;
end if;
t := p + 1 - n;
if nt ne p + 1 + t then
    return false;
end if;
r := n div 8; // "|r| =", Round(Log(2, r));
sec := Log(2, Sqrt(Pi(RealField())*r/4));
"Good Elligator 2 curve: y^2 = x^3 + "*Sprint(A)*"*x^2 + x",
"over GF(2^"*Sprint(m)*" - "*Sprint(delta)*")",
"at sec level 2^"*Sprint(sec),
" with r =", r;
assert IsProbablePrime(r);
```

```
    return true;
end function;

m := 159; A :=  82495;
if not MontyCurve(m, A) then
    "LOGIC ERROR!";
end if;

m := 191; A :=  30327;
if not MontyCurve(m, A) then
    "LOGIC ERROR!";
end if;

m := 221; A := 204400;
if not MontyCurve(m, A) then
    "LOGIC ERROR!";
end if;

m := 255; A := 486662;
if not MontyCurve(m, A) then
    "LOGIC ERROR!";
end if;

m := 383; A := 229969;
if not MontyCurve(m, A) then
    "LOGIC ERROR!";
end if;

function EddieCurve(m, d)
    p := 2^m;
    repeat
        p := PreviousPrime(p);
    until p mod 4 eq 3;
    delta := 2^m - p;
    if delta gt m then
        return false;
    end if;
    F := GF(p);
    e := 1 - d;
    if IsSquare(F!e) or IsSquare(F!d) or not IsSquare(-F!d) then
        return false;
    end if;
    s := Sqrt(-F!d);
    if not IsSquare(2*(s - 1)/(s + 1)) then
        return false;
```

```
    end if;
    // check curve x^2 + y^2 = 1 + dx^2y^2,
    // or equivalently v^2 = u^3 + [(4/e - 2)e]u^2 + [e^2]u:
    ok, E := IsEllipticCurve([0, (4 - 2*F!e), 0, F!e^2, 0]);
    if not ok then
        return false;
    end if;
    n := #E;
    if (n mod 4 ne 0) or not IsProbablePrime(n div 4) then
        return false;
    end if;
    // check twist x^2 + y^2 = 1 + (1/d)x^2y^2,
    // or equivalently v^2 = u^3 + [(4/e - 2)]u^2 + u:
    ok, Et := IsEllipticCurve([0, (4/F!e - 2), 0, 1, 0]);
    if not ok then
        return false;
    end if;
    nt := #Et;
    if (nt mod 4 ne 0) or not IsProbablePrime(nt div 4) then
        return false;
    end if;
    t := p + 1 - n;
    if nt ne p + 1 + t then
        return false;
    end if;
    r := n div 4; // "|r| =", Round(Log(2, r));
    sec := Log(2, Sqrt(Pi(RealField())*r/4));
    "Good Elligator 1 curve: x^2 + y^2 = 1 - "*Sprint(-d)*"*x^2*y^2",
    "over GF(2^"*Sprint(m)*" - "*Sprint(delta)*")",
    "at sec level 2^"*Sprint(sec),
    " with r =", r;
    assert IsProbablePrime(r);
    return true;
end function;

m := 157; d :=  -42000;
if not EddieCurve(m, d) then
    "LOGIC ERROR!";
end if;

m := 191; d :=  -65896;
if not EddieCurve(m, d) then
    "LOGIC ERROR!";
end if;
```

```
m := 226; d :=  -4417;
if not EddieCurve(m, d) then
    "LOGIC ERROR!";
end if;

m := 251; d :=  -1174;
if not EddieCurve(m, d) then
    "LOGIC ERROR!";
end if;

m := 382; d := -67254;
if not EddieCurve(m, d) then
    "LOGIC ERROR!";
end if;
```