Network Working Group
Internet-Draft
Intended status: Informational
Expires: January 4, 2015

B. Black
Microsoft
J. Bos
NXP Semiconductors
C. Costello
P. Longa
M. Naehrig
Microsoft Research

July 3, 2014

# Elliptic Curve Cryptography (ECC) Nothing Up My Sleeve (NUMS) Curves and Curve Generation
## draft-black-numscurves-01

## Abstract

This memo describes a family of deterministically generated Nothing Up My Sleeve (NUMS) elliptic curves over prime fields offering high practical security in cryptographic applications, including Transport Layer Security (TLS) and X.509 certificates. The domain parameters are defined for both classical Weierstrass curves, for compatibility with existing applications, and modern twisted Edwards curves, allowing further efficiency improvements for a given security level.

## Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 4, 2015.

## Copyright Notice

Table of Contents



## 1. Introduction

Since the initial standardization of elliptic curve cryptography (ECC) in [SEC1] there has been significant progress related to both efficiency and security of curves and implementations. Notable examples are algorithms protected against certain side-channel attacks, different 'special' prime shapes which allow faster modular

arithmetic, and a larger set of curve models from which to choose. There is also concern in the community regarding the generation and potential weaknesses of the curves defined in [NIST].

This memo describes a set of elliptic curves for cryptography, defined in [MSR] which have been specifically chosen to support constant-time, exception-free scalar multiplications that are resistant to a wide range of side-channel attacks including timing and cache attacks, thereby offering high practical security in cryptographic applications. These curves are deterministically generated based on algorithms defined in this document and without any hidden parameters or reliance on randomness, hence they are called Nothing Up My Sleeve (NUMS) curves. The domain parameters are defined for both classical Weierstrass curves, for compatibility with existing applications while delivering better performance and stronger security, and modern twisted Edwards curves, allowing even further efficiency improvements for a given security level.

### 1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

## 2. Scope and Relation to Other Specifications

This RFC specifies elliptic curve domain parameters over prime fields GF(p) with p having a length of 256, 384, and 512 bits, in both Weierstrass and twisted Edwards form. These parameters were generated in a transparent and deterministic way and have been shown to resist current cryptanalytic approaches. Furthermore, this document identifies the security and implementation requirements for the parameters, and describes the methods used for the deterministic generation of the parameters.

This document also describes use of the specified parameters in X.509 certificates, in accordance with [RFC3279] and [RFC5480]. It does not address the cryptographic algorithms to be used with the specified parameters nor their application in other standards. However, it is consistent with the following RFCs that specify the usage of ECC in protocols and applications:

- [RFC4050] for XML signatures
- [RFC4492] for TLS
- [RFC4754] for IKE

- o [RFC5753] for cryptographic message syntax (CMS)

# 3. Requirements

## 3.1. Technical Requirements

1. Applicability to multiple cryptographic algorithms without transformation, in particular key exchange, e.g. Elliptic Curve Diffie-Hellman (ECDH), and digital signature algorithms, e.g., (ECDSA), Schnorr.

2. Multiple security levels using the same curve generation algorithm with only a security parameter change. The curve generation algorithm must be extensible to any security level.

3. Ability to use pre-computation for increased performance. In particular, speed-up in key generation is important when a curve is used with ephemeral key exchange algorithm, such as ECDHE.

4. The bit length of prime and order of curves for a given security level MUST be divisible by 8. Specifically, constructions such as NIST P-521 are to be avoided as they introduce interoperability and implementation problems.

## 3.2. Security Requirements

For each curve type (twisted Edwards or Weierstrass) at a specific specific security level:

1. The domain parameters SHALL be generated in a simple, deterministic manner, without any secret or random inputs. The derivation of the curve parameters is defined in Appendix A.

2. The curve SHALL NOT restrict the scalars to a small subset. Using full-set scalars prevents implementation pitfalls that might otherwise go unnoticed.

3. The curve selection SHALL include prime order curves with cofactor 1 only. Composite order curves require changes in protocols and in implementations. Additionally, implementations for composite order curves must thwart subgroup attacks.

4. The trace of Frobenius MUST NOT be in {0, 1} in order to rule out the attacks described in [Smart], [AS], and [S], as in [EBP].

5. MOV Degree: the embedding degree k MUST be greater than (r - 1) / 100, as in [EBP].

6. CM Discriminant: discriminant D MUST be greater than 2^100, as in [SC].

## 4. Notation

Throughout this document, the following notation is used:

- s: Denotes the bit length, here s in {256,384,512}.
- p: Denotes the prime number defining the base field.
- c: A positive integer used in the representation of the prime p = 2^s - c.
- GF(p): The finite field with p elements.
- b: An element in the finite field GF(p), different from -2,2.
- Eb: The elliptic curve Eb/GF(p): y^2 = x^3 - 3x + b in short Weierstrass form, defined over GF(p) by the parameter b.
- rb: The order rb = #Eb(GF(p)) of the group of GF(p)-rational points on Eb.
- tb: The trace of Frobenius tb = p + 1 - rb of Eb.
- rb': The order rb' = #E'b(GF(p)) = p + 1 + tb of the group of GF(p)-rational points on the quadratic twist Eb': y^2 = x^3 - 3x - b.
- d: An element in the finite field GF(p), different from -1,0.
- Ed: The elliptic curve Ed/GF(p): -x^2 + y^2 = 1 + dx^2y^2 in twisted Edwards form, defined over GF(p) by the parameter d.
- rd: The subgroup order such that 4 * rd = #Ed(GF(p)) is the order of the group of GF(p)-rational points on Ed.
- td: The trace of Frobenius td = p + 1 - 4 * rd of Ed.
- rd': The subgroup order such that 4 * rd' = #Ed'(GF(p)) = p + 1 + tb is the order of the group of GF(p)-rational points on the quadratic twist Ed': -x^2 = y^2 = 1 + (1 / d) * x^2 * y^2.
- P: A generator point defined over GF(p) either of prime order rb in the Weierstrass curve Eb, or of prime order rd on the twisted Edwards curve Ed.
- X(P): The x-coordinate of the elliptic curve point P.
- Y(P): The y-coordinate of the elliptic curve point P.

## 5. Curve Parameters

### 5.1. Parameters for 256-bit Curves

```
   p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFF43
   a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFF40
   b = 0x25581
   r = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFE43C8275EA265C60E43C8275E
       A265C60
X(P) = 0x01
Y(P) = 0x696F1853C1E466D7FC82C96CCEEEDD6BD02C2F9375894EC10BF46306C
       2B56C77
   h = 0x01
```

Curve-Id: numsp256d1

```
   p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFF43
   a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFF42
   d = 0x3BEE
   r = 0x3FFFFFFFFFFFFFFFFFFFFFFFFFFFFFBE6AA55AD0A6BC64E5B84E6F1
       122B4AD
X(P) = 0x0D
Y(P) = 0x7D0AB41E2A1276DBA3D330B39FA046BFBE2A6D63824D303F707F6FB53
       31CADBA
   h = 0x04
```

Curve-Id: numsp256t1

## 5.2. Parameters for 384-bit Curves

```
   p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC3
   a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC0
   b = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF77BB
   r = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFD61EAF1EE
       B5D6881BEDA9D3D4C37E27A604D81F67B0E61B9
X(P) = 0x02
Y(P) = 0x3C9F82CB4B87B4DC71E763E0663E5DBD8034ED422F04F82673330DC58
       D15FFA2B4A3D0BAD5D30F865BCBBF503EA66F43
   h = 0x01
```

Curve-Id: numsp384d1

```
   p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC3
   a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFC
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEC2
   d = 0x5158A
   r = 0x3FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFECD7D11ED
       5A259A25A13A0458E39F4E451D6D71F70426E25
X(P) = 0x08
Y(P) = 0x749CDABA136CE9B65BD4471794AA619DAA5C7B4C930BFF8EBD798A8AE
       753C6D72F003860FEBABAD534A4ACF5FA7F5BEE
   h = 0x04
```

Curve-Id: numsp384t1

## 5.3. Parameters for 512-bit Curves

```
   p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFDC7
   a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFFFFFDC4
   b = 0x1D99B
   r = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
       FFFFFFF5B3CA4FB94E7831B4FC258ED97D0BDC63B568B36607CD243CE
       153F390433555D
X(P) = 0x02
Y(P) = 0x1C282EB23327F9711952C250EA61AD53FCC13031CF6DD336E0B932843
       3AFBDD8CC5A1C1F0C716FDC724DDE537C2B0ADB00BB3D08DC83755B20
       5CC30D7F83CF28
   h = 0x01
```

Curve-Id: numsp512d1

```
      p = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFDC7
      a = 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFFFFDC6
      d = 0x9BAA8
      r = 0x3FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
            FFFFFFFA7E50809EFDABBB9A624784F449545F0DCEA5FF0CB800F894E
            78D1CB0B5F0189
   X(P) = 0x20
   Y(P) = 0x7D67E841DC4C467B605091D80869212F9CEB124BF726973F9FF048779
            E1D614E62AE2ECE5057B5DAD96B7A897C1D72799261134638750F4F0C
            B91027543B1C5E
      h = 0x04
```

Curve-Id: numsp512t1

## 6. Object Identifiers and ASN.1 Syntax for X.509 Certificates

### 6.1. Object Identifiers

The root of the tree for the object identifiers defined in this specification is given by:

```
[TBDOID]
```

The following object identifiers represent the domain parameters for the curves defined in this draft:

```
numsp256d1 OBJECT IDENTIFIER ::= {versionOne 1}

numsp256t1 OBJECT IDENTIFIER ::= {versionOne 2}

numsp384d1 OBJECT IDENTIFIER ::= {versionOne 3}

numsp384t1 OBJECT IDENTIFIER ::= {versionOne 4}

numsp512d1 OBJECT IDENTIFIER ::= {versionOne 5}

numsp512t1 OBJECT IDENTIFIER ::= {versionOne 6}
```

### 6.2. ASN.1 Syntax for X.509 Certificates

The domain parameters for the curves specified in this RFC SHALL be used with X.509 certificates according to [RFC5480]. Specifically, the algorithm field of subjectPublicKeyInfo MUST be one of:

- o id-ecPublicKey to indicate that the algorithms that can be used with the subject public key are unrestricted, as required for ECDSA, or

- o id-ecDH to indicate that the algorithm that can be used with the subject public key is restricted to the ECDH key agreement algorithm, or

- o id-ecMQV indicates that the algorithm that can be used with the subject public key is restricted to the Elliptic Curve Menezes-Qu-Vanstone (ECMQV) key agreement algorithm, and

The field algorithm.parameter of subjectPublicKeyInfo MUST be of type namedCurve. No other values for this field are acceptable.

## 7. Acknowledgements

The authors would like to thank Brian Lamacchia and Tolga Acar for their help in the development of this draft.

## 8. Security Considerations

In addition to the discussion in the requirements, [MSR], [SC], and the other reference documents on EC security, users SHOULD match curves with cryptographic functions of similar strength. Specific recommendations for algorithms, per [RFC5480] are as follows:

| Minimum Bits of Security | EC Key Size | Message Digest Algorithm | Curves |
|---|---|---|---|
| 128 | 256 | SHA-256 | numsp256d1/t1 |
| 192 | 384 | SHA-384 | numsp384d1/t1 |
| 256 | 512 | SHA-512 | numsp512d1/t1 |

Table 1

## 9. Intellectual Property Rights

The authors have no knowledge about any intellectual property rights that cover the usage of the domain parameters defined herein. However, readers should be aware that implementations based on these domain parameters may require use of inventions covered by patent rights.

## 10. IANA Considerations

IANA is requested to allocate an object identifier for elliptic curves under the PKIX root declared in [RFC5480]:

```
PKIX1Algorithms2008 { iso(1) identified-organization(3) dod(6)
  internet(1) security(5) mechanisms(5) pkix(7) id-mod(0) 45 }
```

IANA is further requested to allocate object identifiers under this new elliptic curve root for the named curves in Section 6.1.

## 11. References

### 11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

### 11.2. Informative References

[AS] Satoh, T. and K. Araki, "Fermat quotients and the polynomial time discrete log algorithm for anomalous elliptic curves", 1998.

[EBP] ECC Brainpool, "ECC Brainpool Standard Curves and Curve Generation", October 2005, <http://www.ecc-brainpool.org/download/Domain-parameters.pdf>.

[ECCP] Bos, J., Halderman, J., Heninger, N., Moore, J., Naehrig, M., and E. Wustrow, "Elliptic Curve Cryptography in Practice", December 2013, <https://eprint.iacr.org/2013/734>.

[FPPR] Faugere, J., Perret, L., Petit, C., and G. Renault, 2012, <http://dx.doi.org/10.1007/978-3-642-29011-4_4>.

[MSR] Bos, J., Costello, C., Longa, P., and M. Naehrig, "Selecting Elliptic Curves for Cryptography: An Efficiency and Security Analysis", February 2014, <http://eprint.iacr.org/2014/130.pdf>.

[NIST] National Institute of Standards, "Recommended Elliptic Curves for Federal Government Use", July 1999, <http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf>.

[RFC3279] Bassham, L., Polk, W., and R. Housley, "Algorithms and Identifiers for the Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3279, April 2002.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, July 2003.

[RFC4050] Blake-Wilson, S., Karlinger, G., Kobayashi, T., and Y. Wang, "Using the Elliptic Curve Signature Algorithm (ECDSA) for XML Digital Signatures", RFC 4050, April 2005.

[RFC4492] Blake-Wilson, S., Bolyard, N., Gupta, V., Hawk, C., and B. Moeller, "Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS)", RFC 4492, May 2006.

[RFC4754] Fu, D. and J. Solinas, "IKE and IKEv2 Authentication Using the Elliptic Curve Digital Signature Algorithm (ECDSA)", RFC 4754, January 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5480] Turner, S., Brown, D., Yiu, K., Housley, R., and T. Polk, "Elliptic Curve Cryptography Subject Public Key Information", RFC 5480, March 2009.

[RFC5753] Turner, S. and D. Brown, "Use of Elliptic Curve Cryptography (ECC) Algorithms in Cryptographic Message Syntax (CMS)", RFC 5753, January 2010.

[S] Semaev, I., "Evaluation of discrete logarithms on some elliptic curves", 1998.

[SC] Bernstein, D. and T. Lange, "SafeCurves: choosing safe curves for elliptic-curve cryptography", June 2014, <http://safecurves.cr.yp.to/>.

[SEC1] Certicom Research, "SEC 1: Elliptic Curve Cryptography", September 2000, <http://www.secg.org/collateral/sec1_final.pdf>.

[Smart] Smart, N., "The discrete logarithm problem on elliptic curves of trace one", 1999.

## Appendix A. Parameter Generation

This section describes the generation of the curve parameters, namely the base field prime p, the curve parameters b and d for the Weierstrass and twisted Edwards curves, respectively, and a generator point P of the prime order subgroup of the elliptic curve.

### A.1. Prime Generation

For a given bitlength s in {256, 384, 512}, a prime p is selected as a pseudo-Mersenne prime of the form p = 2^s - c for a positive integer c. Each prime is determined by the smallest positive integer c such that p = 2^s - c is prime and p = 3 mod 4.

```
Input: a bit length s in {256, 384, 512}
Output: a prime p = 2^s - c with p = 3 mod 4
  1. Set c = 1
  2. while (p = 2^s - c is not prime) do
       c = c + 4
     end while
  3. Output p
```

GenerateP

### A.2. Deterministic Curve Parameter Generation

#### A.2.1. Weierstrass Curves

For a given bitlength s in {256, 384, 512} and a corresponding prime p = 2^s - c selected according to Section A.1, the elliptic curve Eb in short Weierstrass form is determined by the element b from GF(p), different from -2,2 with smallest absolute value (when represented as an integer in the interval [-(p - 1) / 2, (p - 1) / 2]) such that both group orders rb and rb’ are prime, and the group order rb < p, i.e. tb > 1. In addition, care must be taken to ensure the MOV degree and CM discriminant requirements from Section 3.2 are met.

```
Input: a prime p = 2^s - c with p = 3 mod 4
Output: the parameter b defining the curve Eb
  1. Set b = 1
  2. while (rb is not prime or rb' is not prime) do
       b = b + 1
     end while
  3. if p + 1 < rb then
       b = -b
     end if
  4. Output b
```

GenerateCurveWeierstrass

### A.2.2. Twisted Edwards Curves

For a given bitlength s in {256, 384, 512} and a corresponding prime p = 2^s - c selected according to Section A.1, the elliptic curve Ed in twisted Edwards form is determined by the element d from GF(p), different from -1,0 with smallest value (when represented as a positive integer) such that both subgroup orders rd and rd' are prime, and the group order 4 * rd < p, i.e. td > 1. In addition, care must be taken to ensure the MOV degree and CM discriminant requirements from Section 3.2 are met.

```
Input: a prime p = 2^s - c with p = 3 mod 4
Output: the parameter d defining the curve Ed
  1. Set d = 1
  2. while (rd is not prime or rd' is not prime or 4*rd > p) do
       d = d + 1;
     end while
  3. Output d
```

GenerateCurveTEdwards

## Appendix B. Generators

The generator points on all six curves are selected as the points of order rb and rd, respectively, with the smallest value for x(P) when represented as a positive integer.

```
   Input: a prime p, and a Weierstrass curve parameter b
   Output: a generator point P = (x(P), y(P)) of order rb
   1. Set x = 1
   2. while ((x^3 - 3 * x + b) is not a quadratic residue modulo p) do
        x = x + 1
      end while
   3. Compute an integer s, 0 < s < p, such that
           s^2 = x^3 - 3 * x + b mod p
   4. Set y = min(s, p - s)
   5. Output P = (x, y)
```

GenerateGenWeierstrass

```
   Input: a prime p and a twisted Edwards curve parameter d
   Output: a generator point P = (x(P), y(P)) of order rd
   1. Set x = 1
   2. while ((d * x^2 = 1 mod p)
             or ((1 + x^2) * (1 - d * x^2) is not a quadratic residue
             modulo p)) do x = x + 1
      end while
   3. Compute an integer s, 0 < s < p, such that
           s^2 * (1 - d * x^2) = 1 + x^2 mod p
   4. Set y = min(s, p - s)
   5. Output P = (x, y)
```

GenerateGenTEdwards

## Authors' Addresses

Benjamin Black
Microsoft
One Microsoft Way
Redmond, WA  98115
US

Email: benblack@microsoft.com

Joppe W. Bos
NXP Semiconductors
Interleuvenlaan 80
3001 Leuven
Belgium

Email: joppe.bos@nxp.com

Craig Costello
Microsoft Research
One Microsoft Way
Redmond, WA 98115
US

Email: craigco@microsoft.com

Patrick Longa
Microsoft Research
One Microsoft Way
Redmond, WA 98115
US

Email: plonga@microsoft.com

Michael Naehrig
Microsoft Research
One Microsoft Way
Redmond, WA 98115
US

Email: mnaehrig@microsoft.com