RECOMMENDED ELLIPTIC CURVES FOR FEDERAL GOVERNMENT USE

July 1999

This collection of elliptic curves is recommended for Federal government use and contains choices of private key length and underlying fields.

§1. Parameter Choices

1.1 Choice of Key Lengths

The principal parameters for elliptic curve cryptography are the elliptic curve E and a designated point G on E called the *base point*. The base point has order r, a large prime. The number of points on the curve is $n = fr$ for some integer f (the *cofactor*) not divisible by r. For efficiency reasons, it is desirable to take the cofactor to be as small as possible.

All of the curves given below have cofactors 1, 2, or 4. As a result, the private and public keys are approximately the same length. Each length is chosen to correspond to the cryptovariable length of a common symmetric cryptologic. In each case, the private key length is, at least, approximately twice the symmetric cryptovariable length.

1.2 Choice of Underlying Fields

For each cryptovariable length, there are given two kinds of fields.

- A *prime field* is the field $GF(p)$ which contains a prime number p of elements. The elements of this field are the integers modulo p, and the field arithmetic is implemented in terms of the arithmetic of integers modulo p.

- A *binary field* is the field $GF(2^m)$ which contains 2^m elements for some m (called the *degree* of the field). The elements of this field are the bit strings of length m, and the field arithmetic is implemented in terms of operations on the bits.

The following table gives the sizes of the various underlying fields. By $\|p\|$ is meant the length of the binary expansion of the integer p.

Symmetric CV Length	Example Algorithm	Prime Field	Binary Field
80	SKIPJACK	$\|p\| = 192$	$m = 163$
112	Triple-DES	$\|p\| = 224$	$m = 233$
128	AES Small	$\|p\| = 256$	$m = 283$
192	AES Medium	$\|p\| = 384$	$m = 409$
256	AES Large	$\|p\| = 521$	$m = 571$

1.3 Choice of Basis

To describe the arithmetic of a binary field, it is first necessary to specify how a bit string is to be interpreted. This is referred to as choosing a *basis* for the field. There are two common types of bases: a *polynomial basis* and a *normal basis.*

- A polynomial basis is specified by an irreducible polynomial modulo 2, called the *field polynomial.* The bit string $(a_{m-1} \ \ldots \ a_2 \ a_1 \ a_0)$ is taken to represent the polynomial

$$a_{m-1}\, t^{m-1} + \cdots + a_2\, t^2 + a_1\, t + a_0$$

over $GF(2)$. The field arithmetic is implemented as polynomial arithmetic modulo $p(t)$, where $p(t)$ is the field polynomial.

- A normal basis is specified by an element θ of a particular kind. The bit string $(a_0\ a_1\ a_2\ \ldots\ a_{m-1})$ is taken to represent the element

$$a_0\,\theta + a_1\,\theta^2 + a_2\,\theta^{2^2} + \cdots + a_{m-1}\,\theta^{2^{m-1}}.$$

  Normal basis field arithmetic is not easy to describe or efficient to implement in general, but is for a special class called *Type T low-complexity* normal bases. For a given field degree m, the choice of T specifies the basis and the field arithmetic (see Appendix 2).

There are many polynomial bases and normal bases from which to choose. The following procedures are commonly used to select a basis representation.

- *Polynomial Basis:* If an irreducible *trinomial* $t^m + t^k + 1$ exists over $GF(2)$, then the field polynomial $p(t)$ is chosen to be the irreducible trinomial with the lowest-degree middle term t^k. If no irreducible trinomial exists, then one selects instead a *pentanomial* $t^m+t^a+t^b+t^c+1$. The particular pentanomial chosen has the following properties: the second term t^a has the lowest degree among all irreducible pentanomials of degree m; the third term t^b has the lowest degree among all irreducible pentanomials of degree m and second term t^a; and the fourth term t^c has the lowest degree among all irreducible pentanomials of degree m, second term t^a, and third term t^b.

- *Normal Basis:* Choose the Type T low-complexity normal basis with the smallest T.

For each binary field, the parameters are given for the above basis representations.

1.4 Choice of Curves

Two kinds of curves are given:

- *Pseudo-random* curves are those whose coefficients are generated from the output of a seeded cryptographic hash. If the seed value is given along with the coefficients, it can be verified easily that the coefficients were indeed generated by that method.
- *Special curves* whose coefficients and underlying field have been selected to optimize the efficiency of the elliptic curve operations.

For each size, the following curves are given:

→ A pseudo-random curve over $GF(p)$.

→ A pseudo-random curve over $GF(2^m)$.

→ A special curve over $GF(2^m)$ called a *Koblitz curve* or *anomalous binary curve.*

The pseudo-random curves are generated via the SHA-1 based method given in the ANSI X9.62 and IEEE P1363 standards. (The generation and verification processes are given in Appendices 4 through 7.)

1.5 Choice of Base Points

Any point of order r can serve as the base point. Each curve is supplied with a sample base point $G = (G_x, G_y)$. Users may want to generate their own base points to ensure cryptographic separation of networks.

§2. Curves over Prime Fields

For each prime p, a pseudo-random curve

$$E: \quad y^2 \equiv x^3 - 3\,x + b \pmod{p}$$

of prime order r is listed.[1] (Thus, for these curves, the cofactor is always $f = 1$.) The following parameters are given:

- The prime modulus p
- The order r
- The 160-bit input seed s to the SHA-1 based algorithm
- The output c of the SHA-1 based algorithm
- The coefficient b (satisfying $b^2\,c \equiv -27 \pmod{p}$)
- The base point x coordinate G_x
- The base point y coordinate G_y

The integers p and r are given in decimal form; bit strings and field elements are given in hex.

[1] The selection $a = -3$ for the coefficient of x was made for reasons of efficiency; see IEEE P1363.

Curve P-192

$$p = 62771017353866807638357894232076664160839087\backslash$$
$$00390324961279$$

$$r = 62771017353866807638357894231760590137671947\backslash$$
$$73182842284081$$

$s =$ `3045ae6f c8422f64 ed579528 d38120ea e12196d5`

$c =$ `3099d2bb`
`bfcb2538 542dcd5f b078b6ef 5f3d6fe2 c745de65`

$b =$ `64210519`
`e59c80e7 0fa7e9ab 72243049 feb8deec c146b9b1`

$G_x =$ `188da80e`
`b03090f6 7cbf20eb 43a18800 f4ff0afd 82ff1012`

$G_y =$ `07192b95`
`ffc8da78 631011ed 6b24cdd5 73f977a1 1e794811`

Curve P-224

$p =$ 26959946667150639794667015087019630673557916\
260026308143510066298881

$r =$ 26959946667150639794667015087019625940457807\
714424391721682722368061

$s =$ `bd713447 99d5c7fc dc45b59f a3b9ab8f 6a948bc5`

$c =$ `5b056c7e 11dd68f4`
`0469ee7f 3c7a7d74 f7d12111 6506d031 218291fb`

$b =$ `b4050a85 0c04b3ab`
`f5413256 5044b0b7 d7bfd8ba 270b3943 2355ffb4`

$G_x =$ `b70e0cbd 6bb4bf7f`
`321390b9 4a03c1d3 56c21122 343280d6 115c1d21`

$G_y =$ `bd376388 b5f723fb`
`4c22dfe6 cd4375a0 5a074764 44d58199 85007e34`

Curve P-256

$$p = 115792089210356248762697446949407573530086143415290314195533631308867097853951$$

$$r = 115792089210356248762697446949407573529996955224135760342422259061068512044369$$

$s =$ `c49d3608 86e70493 6a6678e1 139d26b7 819f7e90`

$c =$ `7efba166 2985be94 03cb055c 75d4f7e0 ce8d84a9 c5114abc af317768 0104fa0d`

$b =$ `5ac635d8 aa3a93e7 b3ebbd55 769886bc 651d06b0 cc53b0f6 3bce3c3e 27d2604b`

$G_x =$ `6b17d1f2 e12c4247 f8bce6e5 63a440f2 77037d81 2deb33a0 f4a13945 d898c296`

$G_y =$ `4fe342e2 fe1a7f9b 8ee7eb4a 7c0f9e16 2bce3357 6b315ece cbb64068 37bf51f5`

Curve P-384

$p =$ 39402006196394479212279040100143613805079739\
27046544666794829340424572177149687032904726\
6088258938001861606973112319

$r =$ 39402006196394479212279040100143613805079739\
27046544666794690527962765939911326356939895\
6308152294913554433653942643

$s =$ `a335926a a319a27a 1d00896a 6773a482 7acdac73`

$c =$ `79d1e655 f868f02f`
`ff48dcde e14151dd b80643c1 406d0ca1 0dfe6fc5`
`2009540a 495e8042 ea5f744f 6e184667 cc722483`

$b =$ `b3312fa7 e23ee7e4`
`988e056b e3f82d19 181d9c6e fe814112 0314088f`
`5013875a c656398d 8a2ed19d 2a85c8ed d3ec2aef`

$G_x =$ `aa87ca22 be8b0537`
`8eb1c71e f320ad74 6e1d3b62 8ba79b98 59f741e0`
`82542a38 5502f25d bf55296c 3a545e38 72760ab7`

$G_y =$ `3617de4a 96262c6f`
`5d9e98bf 9292dc29 f8f41dbd 289a147c e9da3113`
`b5f0b8c0 0a60b1ce 1d7e819d 7a431d7c 90ea0e5f`

Curve P-521

$p =$ 686479766013060971498190079908139321726943530\
0143305409394463459185543183397656052122559\
6406614545549772963113914808580371219879997 1\
66438125740282911150571 51

$r =$ 686479766013060971498190079908139321726943530\
0143305409394463459185543183397655394245057\
7463332171975329639963713633211138647686124 4\
03803403728088927070054 49

$s =$ `d09e8800 291cb853 96cc6717 393284aa a0da64ba`

$c =$ `0b4 8bfa5f42`
`0a349495 39d2bdfc 264eeeeb 077688e4 4fbf0ad8`
`f6d0edb3 7bd6b533 28100051 8e19f1b9 ffbe0fe9`
`ed8a3c22 00b8f875 e523868c 70c1e5bf 55bad637`

$b =$ `051 953eb961`
`8e1c9a1f 929a21a0 b68540ee a2da725b 99b315f3`
`b8b48991 8ef109e1 56193951 ec7e937b 1652c0bd`
`3bb1bf07 3573df88 3d2c34f1 ef451fd4 6b503f00`

$G_x =$ `c6 858e06b7`
`0404e9cd 9e3ecb66 2395b442 9c648139 053fb521`
`f828af60 6b4d3dba a14b5e77 efe75928 fe1dc127`
`a2ffa8de 3348b3c1 856a429b f97e7e31 c2e5bd66`

$G_y =$ `118 39296a78`
`9a3bc004 5c8a5fb4 2c7d1bd9 98f54449 579b4468`
`17afbd17 273e662c 97ee7299 5ef42640 c550b901`
`3fad0761 353c7086 a272c240 88be9476 9fd16650`

§3. Curves over Binary Fields

For each field degree m, a pseudo-random curve is given, along with a Koblitz curve. The pseudo-random curve has the form

$$E : y^2 + x\,y = x^3 + \ x^2 + b,$$

and the Koblitz curve has the form

$$E_a : y^2 + x\,y = x^3 + a\,x^2 + 1$$

where $a = 0$ or 1.

For each pseudo-random curve, the cofactor is $f = 2$. The cofactor of each Koblitz curve is $f = 2$ if $a = 1$ and $f = 4$ if $a = 0$.

The coefficients of the pseudo-random curves, and the coordinates of the base points of both kinds of curves, are given in terms of both the polynomial and normal basis representations discussed in §1.3.

For each m, the following parameters are given:

Field Representation:

- The normal basis type T
- The field polynomial (trinomial or pentanomial)

Koblitz Curve:

- The coefficient a
- The base point order r
- The base point x coordinate G_x
- The base point y coordinate G_y

Pseudo-random curve:

- The base point order r

Pseudo-random curve (Polynomial Basis representation):

- The coefficient b
- The base point x coordinate G_x
- The base point y coordinate G_y

Pseudo-random curve (Normal Basis representation):

- The 160-bit input seed s to the SHA-1 based algorithm
- The coefficient b (*i.e.*, the output of the SHA-1 based algorithm)
- The base point x coordinate G_x
- The base point y coordinate G_y

Integers (such as T, m, and r) are given in decimal form; bit strings and field elements are given in hex.

Degree 163 Binary Field

$$T = 4$$

$$p(t) = t^{163} + t^7 + t^6 + t^3 + 1$$

Curve K-163

$$a = \texttt{1}$$

$$r = 5846006549323611672814741753598448348329118574063$$

Polynomial Basis:

$G_x =$ `2 fe13c053 7bbc11ac aa07d793 de4e6d5e 5c94eee8`

$G_y =$ `2 89070fb0 5d38ff58 321f2e80 0536d538 ccdaa3d9`

Normal Basis:

$G_x =$ `0 5679b353 caa46825 fea2d371 3ba450da 0c2a4541`

$G_y =$ `2 35b7c671 00506899 06bac3d9 dec76a83 5591edb2`

Curve B-163

$$r = 5846006549323611672814742442876390689256843201587$$

Polynomial Basis:

$b =$ `2 0a601907 b8c953ca 1481eb10 512f7874 4a3205fd`

$G_x =$ `3 f0eba162 86a2d57e a0991168 d4994637 e8343e36`

$G_y =$ `0 d51fbc6c 71a0094f a2cdd545 b11c5c0c 797324f1`

Normal Basis:

$s =$ `85e25bfe 5c86226c db12016f 7553f9d0 e693a268`

$b =$ `6 645f3cac f1638e13 9c6cd13e f61734fb c9e3d9fb`

$G_x =$ `0 311103c1 7167564a ce77ccb0 9c681f88 6ba54ee8`

$G_y =$ `3 33ac13c6 447f2e67 613bf700 9daf98c8 7bb50c7f`

Degree 233 Binary Field

$$T = 2$$

$$p(t) = t^{233} + t^{74} + 1$$

Curve K-233

$a =$ `0`

$$r = 34508731733952818937173779311385127605709409888622521\backslash$$
$$26328087024741343$$

Polynomial Basis:

$G_x =$ `172 32ba853a 7e731af1`
`29f22ff4 149563a4 19c26bf5 0a4c9d6e efad6126`

$G_y =$ `1db 537dece8 19b7f70f`
`555a67c4 27a8cd9b f18aeb9b 56e0c110 56fae6a3`

Normal Basis:

$G_x =$ `0fd e76d9dcd 26e643ac`
`26f1aa90 1aa12978 4b71fc07 22b2d056 14d650b3`

$G_y =$ `064 3e317633 155c9e04`
`47ba8020 a3c43177 450ee036 d6335014 34cac978`

Curve B-233

$$r = 6901746346790563787434755862277025555839812737345013555379383634485463$$

Polynomial Basis:

$b =$ `066 647ede6c 332c7f8c`
`0923bb58 213b333b 20e9ce42 81fe115f 7d8f90ad`

$G_x =$ `0fa c9dfcbac 8313bb21`
`39f1bb75 5fef65bc 391f8b36 f8f8eb73 71fd558b`

$G_y =$ `100 6a08a419 03350678`
`e58528be bf8a0bef f867a7ca 36716f7e 01f81052`

Normal Basis:

$s =$ `74d59ff0 7f6b413d 0ea14b34 4b20a2db 049b50c3`

$b =$ `1a0 03e0962d 4f9a8e40`
`7c904a95 38163adb 82521260 0c7752ad 52233279`

$G_x =$ `18b 863524b3 cdfefb94`
`f2784e0b 116faac5 4404bc91 62a363ba b84a14c5`

$G_y =$ `049 25df77bd 8b8ff1a5`
`ff519417 822bfedf 2bbd7526 44292c98 c7af6e02`

Degree 283 Binary Field

$$T = 6$$
$$p(t) = t^{283} + t^{12} + t^7 + t^5 + 1$$

Curve K-283

$$a = 0$$
$$r = 3885337784451458141838923813647037813284811733793061\backslash$$
$$24295874997529815829704422603873$$

Polynomial Basis:

$G_x =$ 503213f 78ca4488 3f1a3b81 62f188e5
53cd265f 23c1567a 16876913 b0c2ac24 58492836

$G_y =$ 1ccda38 0f1c9e31 8d90f95d 07e5426f
e87e45c0 e8184698 e4596236 4e341161 77dd2259

Normal Basis:

$G_x =$ 3ab9593 f8db09fc 188f1d7c 4ac9fcc3
e57fcd3b db15024b 212c7022 9de5fcd9 2eb0ea60

$G_y =$ 2118c47 55e7345c d8f603ef 93b98b10
6fe8854f feb9a3b3 04634cc8 3a0e759f 0c2686b1

Curve B-283

$r = 7770675568902916283677847627294075626569625924376904\backslash$
$8910919652677004427778737869287\,1$

Polynomial Basis:

$b =$ `27b680a c8b8596d a5a4af8a 19a0303f`
`ca97fd76 45309fa2 a581485a f6263e31 3b79a2f5`

$G_x =$ `5f93925 8db7dd90 e1934f8c 70b0dfec`
`2eed25b8 557eac9c 80e2e198 f8cdbecd 86b12053`

$G_y =$ `3676854 fe24141c b98fe6d4 b20d02b4`
`516ff702 350eddb0 826779c8 13f0df45 be8112f4`

Normal Basis:

$s =$ `77e2b073 70eb0f83 2a6dd5b6 2dfc88cd 06bb84be`

$b =$ `157261b 894739fb 5a13503f 55f0b3f1`
`0c560116 66331022 01138cc1 80c0206b dafbc951`

$G_x =$ `749468e 464ee468 634b21f7 f61cb700`
`701817e6 bc36a236 4cb8906e 940948ea a463c35d`

$G_y =$ `62968bd 3b489ac5 c9b859da 68475c31`
`5bafcdc4 ccd0dc90 5b70f624 46f49c05 2f49c08c`

Degree 409 Binary Field

$$T = 4$$

$$p(t) = t^{409} + t^{87} + 1$$

Curve K-409

$$a = \texttt{0}$$

$$r = 330527984395124299475957654016385519914202341482140609642324395022880711289249191050673258457777458014096366590617731358671$$

Polynomial Basis:

```
Gx =                   060f05f 658f49c1 ad3ab189
     0f718421 0efd0987 e307c84c 27accfb8 f9f67cc2
     c460189e b5aaaa62 ee222eb1 b35540cf e9023746
Gy =                   1e36905 0b7c4e42 acba1dac
     bf04299c 3460782f 918ea427 e6325165 e9ea10e3
     da5f6c42 e9c55215 aa9ca27a 5863ec48 d8e0286b
```

Normal Basis:

```
Gx =                   1b559c7 cba2422e 3affe133
     43e808b5 5e012d72 6ca0b7e6 a63aeafb c1e3a98e
     10ca0fcf 98350c3b 7f89a975 4a8e1dc0 713cec4a
Gy =                   16d8c42 052f07e7 713e7490
     eff318ba 1abd6fef 8a5433c8 94b24f5c 817aeb79
     852496fb ee803a47 bc8a2038 78ebf1c4 99afd7d6
```

Curve B-409

$$r = 661055968790248598951915308032771039828404682964281219284648798304157774827374805208143723762179110965979867288366567526771$$

Polynomial Basis:

$b =$ 021a5c2 c8ee9feb 5c4b9a75
3b7b476b 7fd6422e f1f3dd67 4761fa99 d6ac27c8
a9a197b2 72822f6c d57a55aa 4f50ae31 7b13545f

$G_x =$ 15d4860 d088ddb3 496b0c60
64756260 441cde4a f1771d4d b01ffe5b 34e59703
dc255a86 8a118051 5603aeab 60794e54 bb7996a7

$G_y =$ 061b1cf ab6be5f3 2bbfa783
24ed106a 7636b9c5 a7bd198d 0158aa4f 5488d08f
38514f1f df4b4f40 d2181b36 81c364ba 0273c706

Normal Basis:

$s =$ 4099b5a4 57f9d69f 79213d09 4c4bcd4d 4262210b

$b =$ 124d065 1c3d3772 f7f5a1fe
6e715559 e2129bdf a04d52f7 b6ac7c53 2cf0ed06
f610072d 88ad2fdc c50c6fde 72843670 f8b3742a

$G_x =$ 0ceacbc 9f475767 d8e69f3b
5dfab398 13685262 bcacf22b 84c7b6dd 981899e7
318c96f0 761f77c6 02c016ce d7c548de 830d708f

$G_y =$ 199d64b a8f089c6 db0e0b61
e80bb959 34afd0ca f2e8be76 d1c5e9af fc7476df
49142691 ad303902 88aa09bc c59c1573 aa3c009a

Degree 571 Binary Field

$$T = 10$$
$$p(t) = t^{571} + t^{10} + t^5 + t^2 + 1$$

Curve K-571

$$a = \texttt{0}$$
$$r = 19322687615086291723476759454659936721494636648532174\backslash$$
$$99328617625725759571144780212268133978522706711834706\backslash$$
$$71280082535146127367497406661731192968242161709250355\backslash$$
$$5733685276673$$

Polynomial Basis:

```
G_x =                     26eb7a8 59923fbc 82189631
      f8103fe4 ac9ca297 0012d5d4 60248048 01841ca4
      43709584 93b205e6 47da304d b4ceb08c bbd1ba39
      494776fb 988b4717 4dca88c7 e2945283 a01c8972
G_y =                     349dc80 7f4fbf37 4f4aeade
      3bca9531 4dd58cec 9f307a54 ffc61efc 006d8a2c
      9d4979c0 ac44aea7 4fbebbb9 f772aedc b620b01a
      7ba7af1b 320430c8 591984f6 01cd4c14 3ef1c7a3
```

Normal Basis:

$G_x =$ `04bb2db a418d0db 107adae0`
`03427e5d 7cc139ac b465e593 4f0bea2a b2f3622b`
`c29b3d5b 9aa7a1fd fd5d8be6 6057c100 8e71e484`
`bcd98f22 bf847642 37673674 29ef2ec5 bc3ebcf7`

$G_y =$ `44cbb57 de20788d 2c952d7b`
`56cf39bd 3e89b189 84bd124e 751ceff4 369dd8da`
`c6a59e6e 745df44d 8220ce22 aa2c852c fcbbef49`
`ebaa98bd 2483e331 80e04286 feaa2530 50caff60`

Curve B-571

$r =$ 386453752301725834469535189093198734429892732970643498\
986572352514515191422895604245361439993894157730831338\
811219269444862468724628168130702345282883033324113931\
3191105285703

Polynomial Basis:

$b =$ `2f40e7e 2221f295 de297117`
`b7f3d62f 5c6a97ff cb8ceff1 cd6ba8ce 4a9a18ad`
`84ffabbd 8efa5933 2be7ad67 56a66e29 4afd185a`
`78ff12aa 520e4de7 39baca0c 7ffeff7f 2955727a`

$G_x =$ `303001d 34b85629 6c16c0d4`
`0d3cd775 0a93d1d2 955fa80a a5f40fc8 db7b2abd`
`bde53950 f4c0d293 cdd711a3 5b67fb14 99ae6003`
`8614f139 4abfa3b4 c850d927 e1e7769c 8eec2d19`

$G_y =$ 37bf273 42da639b 6dccfffe
b73d69d7 8c6c27a6 009cbbca 1980f853 3921e8a6
84423e43 bab08a57 6291af8f 461bb2a8 b3531d2f
0485c19b 16e2f151 6e23dd3c 1a4827af 1b8ac15b

Normal Basis:

$s =$ 2aa058f7 3a0e33ab 486b0f61 0410c53a 7f132310
$b =$ 3762d0d 47116006 179da356
88eeaccf 591a5cde a7500011 8d9608c5 9132d434
26101a1d fb377411 5f586623 f75f0000 1ce61198
3c1275fa 31f5bc9f 4be1a0f4 67f01ca8 85c74777
$G_x =$ 0735e03 5def5925 cc33173e
b2a8ce77 67522b46 6d278b65 0a291612 7dfea9d2
d361089f 0a7a0247 a184e1c7 0d417866 e0fe0feb
0ff8f2f3 f9176418 f97d117e 624e2015 df1662a8
$G_y =$ 04a3642 0572616c df7e606f
ccadaecf c3b76dab 0eb1248d d03fbdfc 9cd3242c
4726be57 9855e812 de7ec5c5 00b4576a 24628048
b6a72d88 0062eed0 dd34b109 6d3acbb6 b01a4a97

Appendix 1: Implementation of Modular Arithmetic

The prime moduli in the above examples are of a special type (called *generalized Mersenne numbers*) for which modular multiplication can be carried out more efficiently than in general. This appendix provides the rules for implementing this faster arithmetic, for each of the prime moduli appearing in the examples.

The usual way to multiply two integers (mod m) is to take the integer product and reduce it (mod m). One therefore has the following problem: given an integer A less than m^2, compute

$$B := A \bmod m.$$

In general, one must obtain B as the remainder of an integer division. If m is a generalized Mersenne number, however, then B can be expressed as a sum or difference (mod m) of a small number of terms. To compute this expression, one can evaluate the integer sum or difference and reduce the result modulo m. The latter reduction can be accomplished by adding or subtracting a few copies of m.

The prime moduli p for each of the five example curves is a generalized Mersenne number.

Curve P-192:

The modulus for this curve is $p = 2^{192} - 2^{64} - 1$. Every integer A less than p^2 can be written

$$A = A_5 \cdot 2^{320} + A_4 \cdot 2^{256} + A_3 \cdot 2^{192} + A_2 \cdot 2^{128} + A_1 \cdot 2^{64} + A_0,$$

where each A_i is a 64-bit integer. The expression for B is

$$B := T + S_1 + S_2 + S_3 \bmod p,$$

where the 192-bit terms are given by

$$\begin{aligned}
T &= A_2 \cdot 2^{128} + A_1 \cdot 2^{64} + A_0 \\
S_1 &= \phantom{A_2 \cdot 2^{128} +{}} A_3 \cdot 2^{64} + A_3 \\
S_2 &= A_4 \cdot 2^{128} + A_4 \cdot 2^{64} \\
S_3 &= A_5 \cdot 2^{128} + A_5 \cdot 2^{64} + A_5.
\end{aligned}$$

Curve P-224:

The modulus for this curve is $p = 2^{224} - 2^{96} + 1$. Every integer A less than p^2 can be written

$$\begin{aligned} A =& A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + A_{10} \cdot 2^{320} + \\ & A_9 \cdot 2^{288} + A_8 \cdot 2^{256} + A_7 \cdot 2^{224} + A_6 \cdot 2^{192} + A_5 \cdot 2^{160} + \\ & A_4 \cdot 2^{128} + A_3 \cdot 2^{96} + A_2 \cdot 2^{64} + A_1 \cdot 2^{32} + A_0, \end{aligned}$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{13} \;\|\; A_{12} \;\|\; \cdots \;\|\; A_0).$$

The expression for B is

$$B := T + S_1 + S_2 - D_1 - D_2 \bmod p,$$

where the 224-bit terms are given by

$$\begin{array}{rccccccccccccccc}
T = (& A_6 & \| & A_5 & \| & A_4 & \| & A_3 & \| & A_2 & \| & A_1 & \| & A_0 &) \\
S_1 = (& A_{10} & \| & A_9 & \| & A_8 & \| & A_7 & \| & 0 & \| & 0 & \| & 0 &) \\
S_2 = (& 0 & \| & A_{13} & \| & A_{12} & \| & A_{11} & \| & 0 & \| & 0 & \| & 0 &) \\
D_1 = (& A_{13} & \| & A_{12} & \| & A_{11} & \| & A_{10} & \| & A_9 & \| & A_8 & \| & A_7 &) \\
D_2 = (& 0 & \| & 0 & \| & 0 & \| & 0 & \| & A_{13} & \| & A_{12} & \| & A_{11} &).
\end{array}$$

Curve P-256:

The modulus for this curve is $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$. Every integer A less than p^2 can be written

$$\begin{aligned} A = & A_{15} \cdot 2^{480} + A_{14} \cdot 2^{448} + A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + \\ & A_{10} \cdot 2^{320} + A_9 \cdot 2^{288} + A_8 \cdot 2^{256} + A_7 \cdot 2^{224} + A_6 \cdot 2^{192} + \\ & A_5 \cdot 2^{160} + A_4 \cdot 2^{128} + A_3 \cdot 2^{96} + A_2 \cdot 2^{64} + A_1 \cdot 2^{32} + A_0, \end{aligned}$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{15} \;\|\; A_{14} \;\|\; \cdots \;\|\; A_0).$$

The expression for B is

$$B := T + 2\,S_1 + 2\,S_2 + S_3 + S_4 - D_1 - D_2 - D_3 - D_4 \bmod p,$$

where the 256-bit terms are given by

$$\begin{array}{lllllllll}
T = (& A_7 & \| \; A_6 & \| \; A_5 & \| \; A_4 & \| \; A_3 & \| \; A_2 & \| \; A_1 & \| \; A_0 \;) \\
S_1 = (& A_{15} & \| \; A_{14} & \| \; A_{13} & \| \; A_{12} & \| \; A_{11} & \| \; 0 & \| \; 0 & \| \; 0 \;) \\
S_2 = (& 0 & \| \; A_{15} & \| \; A_{14} & \| \; A_{13} & \| \; A_{12} & \| \; 0 & \| \; 0 & \| \; 0 \;) \\
S_3 = (& A_{15} & \| \; A_{14} & \| \; 0 & \| \; 0 & \| \; 0 & \| \; A_{10} & \| \; A_9 & \| \; A_8 \;) \\
S_4 = (& A_8 & \| \; A_{13} & \| \; A_{15} & \| \; A_{14} & \| \; A_{13} & \| \; A_{11} & \| \; A_{10} & \| \; A_9 \;) \\
D_1 = (& A_{10} & \| \; A_8 & \| \; 0 & \| \; 0 & \| \; 0 & \| \; A_{13} & \| \; A_{12} & \| \; A_{11} \;) \\
D_2 = (& A_{11} & \| \; A_9 & \| \; 0 & \| \; 0 & \| \; A_{15} & \| \; A_{14} & \| \; A_{13} & \| \; A_{12} \;) \\
D_3 = (& A_{12} & \| \; 0 & \| \; A_{10} & \| \; A_9 & \| \; A_8 & \| \; A_{15} & \| \; A_{14} & \| \; A_{13} \;) \\
D_4 = (& A_{13} & \| \; 0 & \| \; A_{11} & \| \; A_{10} & \| \; A_9 & \| \; 0 & \| \; A_{15} & \| \; A_{14} \;).
\end{array}$$

Curve P-384:

The modulus for this curve is $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$. Every integer A less than p^2 can be written

$$\begin{aligned} A = & A_{23} \cdot 2^{736} + A_{22} \cdot 2^{704} + A_{21} \cdot 2^{672} + A_{20} \cdot 2^{640} + A_{19} \cdot 2^{608} + \\ & A_{18} \cdot 2^{576} + A_{17} \cdot 2^{544} + A_{16} \cdot 2^{512} + A_{15} \cdot 2^{480} + A_{14} \cdot 2^{448} + \\ & A_{13} \cdot 2^{416} + A_{12} \cdot 2^{384} + A_{11} \cdot 2^{352} + A_{10} \cdot 2^{320} + A_{9} \cdot 2^{288} + \\ & A_{8} \cdot 2^{256} + A_{7} \cdot 2^{224} + A_{6} \cdot 2^{192} + A_{5} \cdot 2^{160} + A_{4} \cdot 2^{128} + \\ & A_{3} \cdot 2^{96} + A_{2} \cdot 2^{64} + A_{1} \cdot 2^{32} + A_{0}, \end{aligned}$$

where each A_i is a 32-bit integer. As a concatenation of 32-bit words, this can be denoted by

$$A = (A_{23} \;\|\; A_{22} \;\|\; \cdots \;\|\; A_0).$$

The expression for B is

$$B := T + 2\,S_1 + S_2 + S_3 + S_4 + S_5 + S_6 - D_1 - D_2 - D_3 \bmod p,$$

where the 384-bit terms are given by

$$\begin{aligned}
T &= (A_{11} \| A_{10} \| A_9 \| A_8 \| A_7 \| A_6 \| A_5 \| A_4 \| A_3 \| A_2 \| A_1 \| A_0) \\
S_1 &= (0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| 0 \| 0 \| 0 \| 0) \\
S_2 &= (A_{23} \| A_{22} \| A_{21} \| A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12}) \\
S_3 &= (A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{23} \| A_{22} \| A_{21}) \\
S_4 &= (A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{20} \| 0 \| A_{23} \| 0) \\
S_5 &= (0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| A_{20} \| 0 \| 0 \| 0 \| 0) \\
S_6 &= (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| 0 \| 0 \| A_{20}) \\
D_1 &= (A_{22} \| A_{21} \| A_{20} \| A_{19} \| A_{18} \| A_{17} \| A_{16} \| A_{15} \| A_{14} \| A_{13} \| A_{12} \| A_{23}) \\
D_2 &= (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{22} \| A_{21} \| A_{20} \| 0) \\
D_3 &= (0 \| 0 \| 0 \| 0 \| 0 \| 0 \| 0 \| A_{23} \| A_{23} \| 0 \| 0 \| 0).
\end{aligned}$$

Curve P-521:

The modulus for this curve is $p = 2^{521} - 1$. Every integer A less than p^2 can be written

$$A = A_1 \cdot 2^{521} + A_0,$$

The expression for B is

$$B := A_0 + A_1 \bmod p.$$

Appendix 2: Normal Bases

The elements of $GF(2^m)$ are expressed in terms of the a type T normal basis[2] $\mathcal{B}$ for $GF(2^m)$, for some T. Each element has a unique representation as a bit string

$$(a_0\, a_1 \ldots a_{m-1}).$$

The arithmetic operations are performed as follows.

<u>*Addition:*</u> addition of two elements is implemented by bitwise addition modulo 2. Thus, for example,

$$(1100111) + (1010010) = (0110101).$$

<u>*Squaring:*</u> if

$$\alpha = (a_0\, a_1\, \cdots\, a_{m-1}),$$

then

$$\alpha^2 = (a_{m-1}\, a_0\, a_1\, \cdots\, a_{m-2}).$$

<u>*Multiplication:*</u> to perform multiplication, one first constructs a function $F(\underline{u}, \underline{v})$ on inputs

$$\underline{u} = (u_0\, u_1\, \ldots\, u_{m-1}) \qquad \text{and} \qquad \underline{v} = (v_0\, v_1\, \ldots\, v_{m-1})$$

as follows.

1. Set $p \leftarrow Tm + 1$
2. Let u be an integer having order T modulo p

[2] It is assumed in this section that m is odd and T even, since this is the only case considered in this standard.

3. Compute the sequence $F(1), F(2), \ldots, F(p-1)$ as follows:
   3.1 Set $w \leftarrow 1$
   3.2 For j from 0 to $T-1$ do
      Set $n \leftarrow w$
      For i from 0 to $m-1$ do
         Set $F(n) \leftarrow i$
         Set $n \leftarrow 2n \bmod p$
      Set $w \leftarrow uw \bmod p$
4. Output the formula

$$F(\underline{u}, \underline{v}) := \sum_{k=1}^{p-2} u_{F(k+1)} \, v_{F(p-k)}.$$

This computation need only be performed once per basis.

Given the function F for $\mathcal{B}$, one computes the product

$$(c_0 \, c_1 \ldots c_{m-1}) = (a_0 \, a_1 \ldots a_{m-1}) \times (b_0 \, b_1 \ldots b_{m-1})$$

as follows.

1. Set $(u_0 \, u_1 \, \ldots \, u_{m-1}) \leftarrow (a_0 \, a_1 \, \ldots \, a_{m-1})$
2. Set $(v_0 \, v_1 \, \ldots \, v_{m-1}) \leftarrow (b_0 \, b_1 \, \ldots \, b_{m-1})$
3. For k from 0 to $m-1$ do
   3.1 Compute
   $$c_k := F(\underline{u}, \underline{v})$$
   3.2 Set $u \leftarrow$`LeftShift`(u) and $v \leftarrow$`LeftShift`(v), where `LeftShift` denotes the circular left shift operation.
4. Output $c := (c_0 \, c_1 \, \ldots \, c_{m-1})$

EXAMPLE. For the type 4 normal basis for $GF(2^7)$, one has $p = 29$ and $u = 12$ or 17. Thus the values of F are given by

$F(1) = 0$	$F(8) = 3$	$F(15) = 6$	$F(22) = 5$
$F(2) = 1$	$F(9) = 3$	$F(16) = 4$	$F(23) = 6$
$F(3) = 5$	$F(10) = 2$	$F(17) = 0$	$F(24) = 1$
$F(4) = 2$	$F(11) = 4$	$F(18) = 4$	$F(25) = 2$
$F(5) = 1$	$F(12) = 0$	$F(19) = 2$	$F(26) = 5$
$F(6) = 6$	$F(13) = 4$	$F(20) = 3$	$F(27) = 1$
$F(7) = 5$	$F(14) = 6$	$F(21) = 3$	$F(28) = 0$

Therefore

$$\begin{aligned} F(\underline{u}, \underline{v}) = {} & u_0\, v_1 + u_1\,(v_0 + v_2 + v_5 + v_6) + u_2\,(v_1 + v_3 + v_4 + v_5) \\ & + u_3\,(v_2 + v_5) + u_4\,(v_2 + v_6) + u_5\,(v_1 + v_2 + v_3 + v_6) \\ & + u_6\,(v_1 + v_4 + v_5 + v_6). \end{aligned}$$

Thus, if

$$a = (1\,0\,1\,0\,1\,1\,1) \qquad \text{and} \qquad b = (1\,1\,0\,0\,0\,0\,1),$$

then

$$\begin{aligned} c_0 &= F(\,(1\,0\,1\,0\,1\,1\,1),\,(1\,1\,0\,0\,0\,0\,1)\,) = 1, \\ c_1 &= F(\,(0\,1\,0\,1\,1\,1\,1),\,(1\,0\,0\,0\,0\,1\,1)\,) = 0, \\ &\vdots \\ c_6 &= F(\,(1\,1\,0\,1\,0\,1\,1),\,(1\,1\,1\,0\,0\,0\,0)\,) = 1, \end{aligned}$$

so that $c = ab = (1\,0\,1\,1\,0\,0\,1)$.

Appendix 3: Scalar Multiplication on Koblitz Curves

This appendix describes a particularly efficient method of computing the scalar multiple nP on the Koblitz curve E_a over $GF(2^m)$.

The operation τ is defined by

$$\tau\,(x,y) = (x^2, y^2).$$

When the normal basis representation is used, then the operation τ is implemented by performing right circular shifts on the bit strings representing x and y.

Given m and a, define the following parameters:

- C is some integer greater than 5.

- $\mu := (-1)^{1-a}$

- For $i = 0$ and $i = 1$, define the sequence $s_i(m)$ by

$$s_i(0) = 0, \qquad s_i(1) = 1 - i,$$

$$s_i(m) = \mu \cdot s_i(m-1) - 2\,s_i(m-2) + (-1)^i.$$

- Define the sequence $V(m)$ by

$$V(0) = 2, \qquad V(1) = \mu,$$

$$V(m) = \mu \cdot V(m-1) - 2\,V(m-2).$$

For the example curves, the quantities $s_i(m)$ and $V(m)$ are as follows.

Curve K-163:

$$\begin{aligned}
s_0(163) &= 2579386439110731650419537 \\
s_1(163) &= -755360064476226375461594 \\
V(163) &= -4845466632539410776804317
\end{aligned}$$

Curve K-233:

$$\begin{aligned}
s_0(233) &= -27859711741434429761757834964435883 \\
s_1(233) &= -44192136247082304936052160908934886 \\
V(233) &= -137381546011108235394987299651366779
\end{aligned}$$

Curve K-283:

$$\begin{aligned}
s_0(283) &= -665981532109049041108795536001591469280025 \\
s_1(283) &= 1155860054909136775192281072591609913945968 \\
V(283) &= 7777244870872830999287791970962823977569917
\end{aligned}$$

Curve K-409:

$$\begin{aligned}
s_0(409) &= -18307510456002382137810317198756461378590542487556\backslash \\
&\quad 9338419259 \\
s_1(409) &= -88930485261383040971966532418442126796265661009966\backslash \\
&\quad 06444816790 \\
V(409) &= 10457288737315625927447685387048320737638796957687\backslash \\
&\quad 575791173829
\end{aligned}$$

Curve K-571:

$$s_0(571) = -3737319446876463692429385892476115567147293964596\backslash$$

$$
\begin{aligned}
& 1024123406420235241916729983261305 \\
s_1(571) = & -31918577064464160995381459594895967413196891214856\backslash \\
& 465861056511758982848515832612248752 \\
V(571) = & -148380926981691413899619140297051490364542574180493\backslash \\
& 9362329123395342085168289731114598 43
\end{aligned}
$$

The following algorithm computes the scalar multiple nP on the Koblitz curve E_a over $GF(2^m)$. The average number of elliptic additions and subtractions is at most $\sim 1 + (m/3)$, and is at most $\sim m/3$ with probability at least $1 - 2^{5-C}$.

For $i = 0$ to 1 do

$\quad n' \leftarrow \lfloor n \,/\, 2^{a-C+(m-9)/2} \rfloor$

$\quad g' \leftarrow s_i(m) \cdot n'$

$\quad h' \leftarrow \lfloor g' \,/\, 2^m \rfloor$

$\quad j' \leftarrow V(m) \cdot h'$

$\quad \ell' \leftarrow$ Round$\left((g' + j') \,/\, 2^{(m+5)/2}\right)$

$\quad \lambda_i \leftarrow \ell' \,/\, 2^C$

$\quad f_i \leftarrow$ Round(λ_i)

$\quad \eta_i \leftarrow \lambda_i - f_i$

$\quad h_i \leftarrow 0$

$\eta \leftarrow 2\,\eta_0 + \mu\,\eta_1$

If $\eta \geq 1$

$\quad$ then

$\quad\quad$ if $\eta_0 - 3\,\mu\,\eta_1 < -1$

$\quad\quad\quad$ then set $h_1 \leftarrow \mu$

$\quad\quad\quad$ else set $h_0 \leftarrow 1$

```
    else
        if η0 + 4 μ η1 ≥ 2
            then set h1 ← μ
If η < −1
    then
        if η0 − 3 μ η1 ≥ 1
            then set h1 ← −μ
            else set h0 ← −1
    else
        if η0 + 4 μ η1 < −2
            then set h1 ← −μ
q0 ← f0 + h0
q1 ← f1 + h1
r0 ← n − (s0 + μ s1) q0 − 2 s1 q1
r1 ← s1 q0 − s0 q1
Set Q ← O
P0 ← P
While r0 ≠ 0 or r1 ≠ 0
    If r0 odd then
        set u ← 2 − (r0 − 2 r1 mod 4)
        set r0 ← r0 − u
        if u = 1 then set Q ← Q + P0
        if u = −1 then set Q ← Q − P0
    Set P0 ← τP0
    Set (r0, r1) ← (r1 + μ r0/2, −r0/2)
EndWhile
Output Q
```

Appendix 4: Generation of Pseudo-Random Curves (Prime Case)

Let ℓ be the bit length of p, and define

$$v = \lfloor (\ell - 1)/160 \rfloor$$
$$w = \ell - 160v - 1$$

1. Choose an arbitrary 160-bit string s.
2. Compute h :=SHA-1(s).
3. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
4. Let z be the integer whose binary expansion is given by the 160-bit string s.
5. For i from 1 to v do:
   5.1 Define the 160-bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{160})$.
   5.2 Compute h_i :=SHA-1(s_i).
6. Let h be the bit string obtained by the concatenation of $h_0, h_1, \ldots, h_v$ as follows:

$$h = h_0 \| h_1 \| \ldots \| h_v.$$

7. Let c be the integer whose binary expansion is given by the bit string h.
8. If $c = 0$ or $4c + 27 \equiv 0 \pmod{p}$, then go to Step 1.
9. Choose integers $a, b \in GF(p)$ such that

$$c\, b^2 \equiv a^3 \pmod{p}.$$

   (The simplest choice is $a = c$ and $b = c$. However, one may want to choose differently for performance reasons.)
10. Check that the elliptic curve E over $GF(p)$ given by $y^2 = x^3 + ax + b$ has suitable order. If not, go to Step 1.

Appendix 5: Verification of Curve Pseudo-Randomness (Prime Case)

Given the 160-bit seed value s, one can verify that the coefficient b was obtained from s via the cryptographic hash function SHA-1 as follows. Let ℓ be the bit length of p, and define

$$v = \lfloor (\ell - 1)/160 \rfloor$$

$$w = \ell - 160v - 1$$

1. Compute h :=SHA-1(s).
2. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
3. Let z be the integer whose binary expansion is given by the 160-bit string s.
4. For i from 1 to v do
   4.1 Define the 160-bit string s_i to be binary expansion of the integer $(z + i) \bmod (2^{160})$
   4.2 Compute h_i :=SHA-1(s_i).
5. Let h be the bit string obtained by the concatenation of h_0, $h_1, \ldots,$ h_v as follows:

$$h = h_0 \| h_1 \| \ldots \| h_v.$$

6. Let c be the integer whose binary expansion is given by the bit string h.
7. Verify that $b^2\, c \equiv -27 \pmod{p}$.

Appendix 6: Generation of Pseudo-Random Curves (Binary Case)

Let:

$$v = \lfloor (m-1)/B \rfloor$$
$$w = m - B\,v$$

1. Choose an arbitrary 160-bit string s.
2. Compute h :=SHA-1(s).
3. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
4. Let z be the integer whose binary expansion is given by the 160-bit string s.
5. For i from 1 to v do:
   - 5.1 Define the 160-bit string s_i to be binary expansion of the integer $(z+i) \bmod (2^{160})$.
   - 5.2 Compute h_i :=SHA-1(s_i).
6. Let h be the bit string obtained by the concatenation of h_0, $h_1, \ldots,$ h_v as follows:

$$h = h_0 \| h_1 \| \ldots \| h_v .$$

7. Let b be the element of $GF(2^m)$ whose binary expansion is given by the bit string h.
8. Choose an element a of $GF(2^m)$.
9. Check that the elliptic curve E over $GF(2^m)$ given by $y^2 + xy = x^3 + ax^2 + b$ has suitable order. If not, go to Step 1.

Appendix 7: Verification of Curve Pseudo-Randomness (Binary Case)

Given the 160-bit seed value s, one can verify that the coefficient b was obtained from s via the cryptographic hash function SHA-1 as follows. Define

$$v = \lfloor (m-1)/160 \rfloor$$

$$w = m - 160v$$

1. Compute $h :=$SHA-1(s).
2. Let h_0 be the bit string obtained by taking the w rightmost bits of h.
3. Let z be the integer whose binary expansion is given by the 160-bit string s.
4. For i from 1 to v do
   4.1 Define the 160-bit string s_i to be binary expansion of the integer $(z+i) \bmod (2^{160})$
   4.2 Compute $h_i :=$SHA-1(s_i).
5. Let h be the bit string obtained by the concatenation of h_0, $h_1, \ldots,$ h_v as follows:

$$h = h_0 \| h_1 \| \ldots \| h_v.$$

6. Let c be the element of $GF(2^m)$ which is represented by the bit string h.
7. Verify that $c = b$.

Appendix 8: Polynomial Basis to Normal Basis Conversion

Suppose that α an element of the field $GF(2^m)$. Denote by $\mathbf{p}$ the bit string representing α with respect to a given polynomial basis. It is desired to compute $\mathbf{n}$, the bit string representing α with respect to a given normal basis. This is done via the matrix computation

$$\mathbf{p}\,\Gamma = \mathbf{n},$$

where Γ is an m-by-m matrix with entries in $GF(2)$. The matrix Γ, which depends only on the bases, can be computed easily given its second-to-last row. The second-to-last row for each conversion is given in the table below.

Degree 163:

```
3 e173bfaf 3a86434d 883a2918 a489ddbd 69fe84e1
```

Degree 233:

```
                                0be 19b89595 28bbc490
 038f4bc4 da8bdfc1 ca36bb05 853fd0ed 0ae200ce
```

Degree 283:

```
                      3347f17 521fdabc 62ec1551 acf156fb
 0bceb855 f174d4c1 7807511c 9f745382 add53bc3
```

Degree 409:

```
                               0eb00f2 ea95fd6c 64024e7f
 0b68b81f 5ff8a467 acc2b4c3 b9372843 6265c7ff
 a06d896c ae3a7e31 e295ec30 3eb9f769 de78bef5
```

Degree 571:

```
                                    7940ffa ef996513 4d59dcbf
e5bf239b e4fe4b41 05959c5d 4d942ffd 46ea35f3
e3cdb0e1 04a2aa01 cef30a3a 49478011 196bfb43
c55091b6 1174d7c0 8d0cdd61 3bf6748a bad972a4
```

Given the second-to-last row $\mathbf{r}$ of Γ, the rest of the matrix is computed as follows. Let β be the element of $GF(2^m)$ whose representation with respect to the normal basis is $\mathbf{r}$. Then the rows of Γ, from top to bottom, are the bit strings representing the elements

$$\beta^{m-1},\ \beta^{m-2},\ \ldots,\ \beta^2,\ \beta,\ 1$$

with respect to the normal basis. (Note that the element 1 is represented by the all-1 bit string.)

Alternatively, the matrix is the inverse of the matrix described in Appendix 9.

More details of these computations can be found in Annex A.7 of the IEEE P1363 standard.

Appendix 9: Normal Basis to Polynomial Basis Conversion

Suppose that α an element of the field $GF(2^m)$. Denote by $\mathbf{n}$ the bit string representing α with respect to a given normal basis. It is desired to compute $\mathbf{p}$, the bit string representing α with respect to a given polynomial basis. This is done via the matrix computation

$$\mathbf{n}\,\Gamma = \mathbf{p},$$

where Γ is an m-by-m matrix with entries in $GF(2)$. The matrix Γ, which depends only on the bases, can be computed easily given its top row. The top row for each conversion is given in the table below.

Degree 163:

```
7 15169c10 9c612e39 0d347c74 8342bcd3 b02a0bef
```

Degree 233:

```
                                     149 9e398ac5 d79e3685
59b35ca4 9bb7305d a6c0390b cf9e2300 253203c9
```

Degree 283:

```
         31e0ed7 91c3282d c5624a72 0818049d
053e8c7a b8663792 bc1d792e ba9867fc 7b317a99
```

Degree 409:

```
                  0dfa06b e206aa97 b7a41fff
b9b0c55f 8f048062 fbe8381b 4248adf9 2912ccc8
e3f91a24 e1cfb395 0532b988 971c2304 2e85708d
```

Degree 571:

```
                    452186b bf5840a0 bcf8c9f0
2a54efa0 4e813b43 c3d41496 06c4d27b 487bf107
393c8907 f79d9778 beb35ee8 7467d328 8274caeb
da6ce05a eb4ca5cf 3c3044bd 4372232f 2c1a27c4
```

Given the top row $\mathbf{r}$ of Γ, the rest of the matrix is computed as follows. Let β be the element of $GF(2^m)$ whose representation with respect to the polynomial basis is $\mathbf{r}$. Then the rows of Γ, from top to bottom, are the bit strings representing the elements

$$\beta,\ \beta^2,\ \beta^{2^2},\ \ldots,\ \beta^{2^{m-1}}$$

with respect to the polynomial basis.

Alternatively, the matrix is the inverse of the matrix described in Appendix 8.

More details of these computations can be found in Annex A.7 of the IEEE P1363 standard.