The Digital Signature Scheme ECGDSA

Erwin Hess, Marcus Schafheutle, and Pascale Serf
Siemens AG
Corporate Technology
Dept. CT IC 3

October 24, 2006

Contents

1 Introduction

The purpose of this document is to provide a reference of the ECGDSA digital signature algorithm over finite prime fields $\mathrm{GF}(p)$ and finite extension fields $\mathrm{GF}(2^n)$.

A generic description of the algorithm is given together with various examples of how the ECGDSA scheme works in combination with the hash functions RIPEMD-160 [4], SHA-1 (= SHA-160), SHA-224, SHA-256, SHA-384, and SHA-512 [1]. The elliptic curves used in these examples are taken from [3].

The ECGDSA signature scheme over $\mathrm{GF}(p)$ was developed in 1990, hence, it is actually older than the well known signature schemes ECDSA and DSA. Two ideas were behind the ECGDSA development: Firstly, to transfer T. ElGamal's [5] concept of a digital signature scheme based on the discrete logarithm problem in the multiplicative group of some finite field to elliptic curves. Secondly, to run the new scheme on RSA hardware — existing or just under development. As these hardware implementations did not support fast modular division, the scheme makes use of the modification suggested by Agnew, Mullin, and Vanstone [2] avoiding modular inversions for the calculation of ephemeral keys.

The ECGDSA is also described in ISO/IEC 15946-2: Information technology – Security techniques – Cryptographic techniques based on elliptic curves – Part 2: Digital signatures.[1]

We are not aware of anyone claiming patents covering this algorithm.[2]

Note that this document is just a description of the signature algorithm and does not treat the problem of efficient and secure implementations. Also, this document does not make any statement about the actual security level of the included combinations of curve parameters and hash functions.[3]

This document is organized as follows: In chapter 2 we give the description of the ECGDSA algorithm defined over finite prime fields. Chapter 3 deals with the ECGDSA algorithm defined over finite extension fields of type $\mathrm{GF}(2^n)$. Finally, in chapter 4 the ASN.1 specification of the ECGDSA is given.

---

[1] The currently used name ECGDSA for this algorithm was introduced by the ISO working group SC 27 during the development of the ISO/IEC 15946 standard as an acronym for **E**lliptic **C**urve based **G**erman **D**igital **S**ignature **A**lgorithm reflecting the fact that the the algorithm was contributed by the German national SC 27 body. Analogously, a Korean elliptic curve based signature scheme was denoted ECKDSA.

[2] The ECGDSA was developed at Siemens Corporate Technology where the algorithm was considered a straightforward consequence of the basic ideas described mainly in T. ElGamal [5], V. Miller [12], and N. Koblitz [10]. Therefore, no attempt was made to apply for a patent. Consequently, the ECGDSA is not covered by any Siemens patents.

[3] SHA-1 was recently reported to have weaknesses and thus should not be used for new products. It was included in this document just for compatibility reasons.

Notation

The following notation will be used:

p: a prime number

GF(p): the finite field with the p elements $0, 1, \ldots, p-1$

n: a positive integer

$f(x)$: an irreducible polynomial of degree n over GF(2), where GF(2) is the finite field with 2 elements

GF(2^n): the finite field with 2^n elements

E: an elliptic curve over GF(p) or over GF(2^n). In the case of a curve over GF(p), the elliptic curve E is given by a Weierstrass equation

$$E: \; y^2 = x^3 + ax + b \qquad \text{with } a, b \in \mathrm{GF}(p).$$

In the case of an elliptic curve over GF(2^n), E is given by a Weierstrass equation

$$E: \; y^2 + xy = x^3 + ax^2 + b \qquad \text{with } a, b \in \mathrm{GF}(2^n)$$

$E(\mathrm{GF}(p))$: the group of points of E over GF(p), i. e., the points on E with coordinates in GF(p), including the point at infinity

$E(\mathrm{GF}(2^n))$: the group of points of E over GF(2^n), i. e., the points on E with coordinates in GF(2^n), including the point at infinity

$\#E(\mathrm{GF}(p))$: the cardinality, i. e., the number of elements, of $E(\mathrm{GF}(p))$

$\#E(\mathrm{GF}(2^n))$: the cardinality, i. e., the number of elements, of $E(\mathrm{GF}(2^n))$

q: a large prime divisor of the cardinality of $\#E(\mathrm{GF}(p))$ or of $\#E(\mathrm{GF}(2^n))$, respectively. (Good cryptographic curves must be chosen in such a way that such a q exists.)

$P + Q$: the sum of two points P and Q on E

$k \cdot P$: the k-th multiple of the point P on E, i. e., $\underbrace{P + \cdots + P}_{k \text{ times}}$

$x(P)$:	the x-coordinate of the point P on E
$y(P)$:	the y-coordinate of the point P on E
G:	a point on E over GF(p) (or over GF(2^n), respectively) with order q, i. e., the multiples of G form a subgroup of $E(\mathrm{GF}(p))$ (or of $E(\mathrm{GF}(2^n))$, rspectively) with exactly q elements
m:	the message to be signed
h:	a hash function, mapping the message m to the hash value $h(m)$ with $0 \leq h(m) < q$ (see Remark 1)
$\pi(P)$:	the non-negative integer obtained from the point P on E by the conversion or projection function $\pi(\,)$, explained in Remark 2

Remark 1: We write the hash value in hexadecimal notation, in 32-bit blocks, and interprete

$$h_{8l+7} \ldots h_{8l+1} h_{8l} \qquad \ldots \qquad h_{23} \ldots h_{17} h_{16} \qquad h_{15} \ldots h_9 h_8 \qquad h_7 \ldots h_1 h_0,$$

where the h_i for $i = 0, \ldots, 8l+7$ are hexadecimal digits, as the integer

$$h_{8l+7} \cdot 16^{8l+7} \;+\; \ldots \;+\; h_1 \cdot 16^1 \;+\; h_0 \cdot 16^0,$$

i. e., we use big endian notation.

Remark 2: ISO/IEC 15946-2 uses the general projection function $\pi(\,)$ in the description of ECGDSA. For elliptic curves over GF(p), $\pi(\,)$ equals $x(\,)$, i. e., it maps each point on the elliptic curve to its x-coordinate. We will write explicitly $x(\,)$ instead of $\pi(\,)$, to make things easier.

For elliptic curves over GF(2^n), $\pi(\,)$ is the following conversion function: Let $P = (x(P), y(P))$ be a point on E and $b_{n-1} \ldots b_1 b_0$ the bit sequence representing the field element $x(P)$ of GF(2^n). Then $\pi(P)$ is the integer

$$\pi(P) \;=\; b_{n-1} \cdot 2^{n-1} + \cdots + b_1 \cdot 2 + b_0.$$

2 The digital signature scheme ECGDSA over $\mathrm{GF}(p)$

In this chapter we give a description and provide a reference of the ECGDSA over finite prime fields.

2.1 Private and public key

The private key of the signer A is a randomly chosen integer

$$d_A \in \{1, \ldots, q-1\}.$$

The corresponding public key of A is the point

$$P_A = (d_A^{-1} \bmod q) \cdot G.$$

2.2 Signature generation

In order to sign the message m with the private key d_A, the signer A performs the following steps:

1) A computes the hash value $h(m)$, $0 \leq h(m) < q$.

2) A chooses a random integer $k \in \{1, \ldots, q-1\}$.

3) A determines

$$r = x(k \cdot G) \bmod q.$$

   If $r = 0$, A goes back to step 2) and chooses a new random k.

4) A computes the value

$$s = (k \cdot r - h(m)) \cdot d_A \bmod q.$$

   If $s = 0$, A goes back to step 2) and chooses a new random k.

The pair (r, s) is A's signature of the message m.

2.3 Signature verification

In order to verify whether a pair (r, s) is A's signature of the message m, the verifier B performs the following steps, using A's public key P_A:

1) B checks whether r and s are in $\{1, \ldots, q-1\}$.
   If not, (r, s) is not accepted as A's signature of the message m.

2) B computes the hash value $h(m) < q$.

3) B computes the value

$$u_1 = r^{-1} \cdot h(m) \bmod q.$$

4) B computes the value

$$u_2 = r^{-1} \cdot s \bmod q.$$

5) B determines

$$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q.$$

   If — and only if — this value equals r, i. e., $x(u_1 \cdot G + u_2 \cdot P_A) = r \bmod q$, the pair (r, s) is accepted as A's signature of the message m.

2.4 Examples

For the examples, we will use the elliptic curves `brainpoolP192r1`, `brainpoolP256r1`, `brainpoolP320r1`, `brainpoolP384r1`, and `brainpoolP512r1`, where the order q of the point G has 192, 256, 320, 384, and 512 bits, respectively. The `brainpool` curves are specified in [3].

2.4.1 Examples of ECGDSA over $\mathrm{GF}(p)$ with the hash function RIPEMD-160

Example of ECGDSA over $\mathrm{GF}(p)$ with the 192-bit elliptic curve `brainpoolP192r1` and the hash function RIPEMD-160

For `brainpoolP192r1`, the underlying prime p is

$$p = \texttt{C302F41D 932A36CD A7A34630 93D18DB7 8FCE476D E1A86297}$$

with 192 bits. The elliptic curve $E: y^2 = x^3 + ax + b$ is given by

$$a = \texttt{6A911740 76B1E0E1 9C39C031 FE8685C1 CAE040E5 C69A28EF}$$

and

$$b = \texttt{469A28EF 7C28CCA3 DC721D04 4F4496BC CA7EF414 6FBF25C9}.$$

Its cardinality is

$$\#E(\mathrm{GF}(p)) = q,$$

where

$$q = \texttt{C302F41D 932A36CD A7A3462F 9E9E916B 5BE8F102 9AC4ACC1}$$

is a 192-bit prime. $G = (x(G), y(G))$ with

$$x(G) = \texttt{C0A0647E AAB6A487 53B033C5 6CB0F090 0A2F5C48 53375FD6}$$

and

$$y(G) = \texttt{14B69086 6ABD5BB8 8B5F4828 C1490002 E6773FA2 FA299B8F}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$d_A = \texttt{80F2425E 89B4F585 F27F3536 ED834D68 E3E492DE 08FE84B9}.$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{BCAD67EA E3563528 FEDCBDD8 FC5DA1EE 64123AE0 8BD476B0}$$

and

$$y(P_A) = \texttt{A9ED7D6B 7B9D2929 5DEA48BA 01D3C8B5 6E736885 22A28A04}.$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{22C17C2A 367DD85A B8A365ED 06F19C43 F9ED1834 9A9BC044}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{2D017BE7 F117FF99 4ED6FC63 CA5B4C7A 0430E9FA 095DAFC4}.$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{C02B5CC5 C51D5411 060BF024 5049F824 839F671D 78A1BBF1}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{06664D48 33E54C21 58B4275E D63DE697 B8101E9B C5718A8A}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{240B83FE 9A1DA756 D2C68A06 43EC2052 74F085A6 BFA868D2}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

$$\texttt{2D017BE7 F117FF99 4ED6FC63 CA5B4C7A 0430E9FA 095DAFC4},$$

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 256-bit elliptic curve `brainpoolP256r1` and the hash function RIPEMD-160

256-bit numbers are represented by eight 32-bit blocks. We write the four high significant 32-bit blocks in the first line and the four low significant 32-bit blocks in the second line.

For `brainpoolP256r1`, the underlying prime p is

$$\begin{aligned} p &= \texttt{A9FB57DB A1EEA9BC 3E660A90 9D838D72} \\ & \texttt{6E3BF623 D5262028 2013481D 1F6E5377} \end{aligned}$$

with 256 bits. The elliptic curve $E:\ y^2 = x^3 + ax + b$ is given by

$$\begin{aligned} a &= \texttt{7D5A0975 FC2C3057 EEF67530 417AFFE7} \\ & \texttt{FB8055C1 26DC5C6C E94A4B44 F330B5D9} \end{aligned}$$

and

$$\begin{aligned} b &= \texttt{26DC5C6C E94A4B44 F330B5D9 BBD77CBF} \\ & \texttt{95841629 5CF7E1CE 6BCCDC18 FF8C07B6}. \end{aligned}$$

Its cardinality is

$$\#E(\mathrm{GF}(p)) \ = \ q,$$

where

$$\begin{aligned} q &= \texttt{A9FB57DB A1EEA9BC 3E660A90 9D838D71} \\ & \texttt{8C397AA3 B561A6F7 901E0E82 974856A7} \end{aligned}$$

is a 256-bit prime. $G = (x(G), y(G))$ with

$$\begin{aligned} x(G) &= \texttt{8BD2AEB9 CB7E57CB 2C4B482F FC81B7AF} \\ & \texttt{B9DE27E1 E3BD23C2 3A4453BD 9ACE3262} \end{aligned}$$

and

$$\begin{aligned} y(G) &= \texttt{547EF835 C3DAC4FD 97F8461A 14611DC9} \\ & \texttt{C2774513 2DED8E54 5C1D54C7 2F046997} \end{aligned}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$\begin{aligned} d_A &= \texttt{47B3A278 62DEF037 49ACF0D6 00E69F9B} \\ & \texttt{851D01ED AEFA531F 4D168E78 7307F4D8}. \end{aligned}$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{A26A358B D871FDFB 026D7FCE 6E90B894}$$
$$\texttt{A96EE61A 8938D07D 34E613A1 F78E6A12}$$

and

$$y(P_A) = \texttt{9553E5A3 872CF2FB 02A974B7 F38126AE}$$
$$\texttt{8B6B27D5 F3A2F470 7172B78F C8AD874E}.$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 577EF842}$$
$$\texttt{B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{908E3099 776261A4 558FF7A9 FA6DFFE0}$$
$$\texttt{CA6BB3F9 CB35C2E4 E1DC73FD 5E8C08A3}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{62CCD1D2 91E62F6A 4FFBD966 C66C85AA}$$
$$\texttt{BA990BB6 AB0C087D BD54A456 CCC84E4C}.$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{9119719B 08EEA0D6 BC56E4D1 D37369BC}$$
$$\texttt{F3768445 EF65CAE4 A37BF6D4 3BD01646}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 577EF842}$$
$$\texttt{B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{381596B9 058C22C9 73A255D9 9CA3A046}$$
$$\texttt{4367D62D D9D5DF36 71E80EC2 88CA2595}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{421B839E 82312607 2E43CC2C DA8A0E3B} \\ \texttt{2161C8C7 0DAA8CAA 75F918FA 8C77B3D4}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
62CCD1D2 91E62F6A 4FFBD966 C66C85AA
BA990BB6 AB0C087D BD54A456 CCC84E4C,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 320-bit elliptic curve `brainpoolP320r1` and the hash function RIPEMD-160

320-bit numbers are represented by ten 32-bit blocks. We write the five high significant 32-bit blocks in the first line and the five low significant 32-bit blocks in the second line.

For `brainpoolP320r1`, the underlying prime p is

$$\begin{aligned} p \;&=\; \texttt{D35E4720 36BC4FB7 E13C785E D201E065 F98FCFA6} \\ &\;\; \texttt{F6F40DEF 4F92B9EC 7893EC28 FCD412B1 F1B32E27} \end{aligned}$$

with 320 bits. The elliptic curve $E:\ y^2 = x^3 + ax + b$ is given by

$$\begin{aligned} a \;&=\; \texttt{3EE30B56 8FBAB0F8 83CCEBD4 6D3F3BB8 A2A73513} \\ &\;\; \texttt{F5EB79DA 66190EB0 85FFA9F4 92F375A9 7D860EB4} \end{aligned}$$

and

$$\begin{aligned} b \;&=\; \texttt{52088394 9DFDBC42 D3AD1986 40688A6F E13F4134} \\ &\;\; \texttt{9554B49A CC31DCCD 88453981 6F5EB4AC 8FB1F1A6}. \end{aligned}$$

Its cardinality is

$$\#E(\mathrm{GF}(p)) \;=\; q,$$

where

$$\begin{aligned} q \;&=\; \texttt{D35E4720 36BC4FB7 E13C785E D201E065 F98FCFA5} \\ &\;\; \texttt{B68F12A3 2D482EC7 EE8658E9 8691555B 44C59311} \end{aligned}$$

is a 320-bit prime. $G = (x(G), y(G))$ with

$$\begin{aligned} x(G) \;&=\; \texttt{43BD7E9A FB53D8B8 5289BCC4 8EE5BFE6 F20137D1} \\ &\;\; \texttt{0A087EB6 E7871E2A 10A599C7 10AF8D0D 39E20611} \end{aligned}$$

and

$$\begin{aligned} y(G) \;&=\; \texttt{14FDD055 45EC1CC8 AB409324 7F77275E 0743FFED} \\ &\;\; \texttt{117182EA A9C77877 AAAC6AC7 D35245D1 692E8EE1} \end{aligned}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$\begin{aligned} d_A \;&=\; \texttt{48683594 5A3A284F FC52629A D48D8F37 F4B2E993} \\ &\;\; \texttt{9C52BC72 362A9961 40192AEF 7D2AAFF0 C73A51C5}. \end{aligned}$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{23FF1E03 EC4EBE26 E7F88803 570D5518 EDFF4325}$$
$$\texttt{424D43D4 064B4E8D EEE0356E 19DD6417 449578F2}$$

and

$$y(P_A) = \texttt{5F5D318A B2A492FC E0F5CCF6 C929D1D3 B5CD64FF}$$
$$\texttt{DB53ADD5 E7B4D25A 3993CA3F CE48A7A7 D55DA512.}$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 00000000 00000000}$$
$$\texttt{577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF.}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{C70BC00A 77AD7872 5D36CEEC 27D6F956 FB546EEF}$$
$$\texttt{6DC90E35 31452BD8 7ECE8A4A 7AD730AD C299D81B.}$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4}$$
$$\texttt{FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6.}$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{06AB5250 B31A8E93 56194894 61733200 E4FD5C12}$$
$$\texttt{75C0AB37 E7E41149 5BAAE145 41DF6DE6 66B8CA56.}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 00000000 00000000}$$
$$\texttt{577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF.}$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{95B32CDD E544DB7F 30E27439 3F65D796 C4F5D048}$$
$$\texttt{8A4BCD8B DE6FBFA5 080AE147 2C40E09F B68AF8F5.}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{C11C52F1 DB6B9B1F 3504D1E8 F302EAEA 46061F80}$$
$$\texttt{6C97AB1D 5B4744D9 3ADA3385 23A780E2 23D62980}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4
FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

2.4.2 Examples of ECGDSA over GF(p) with the hash function SHA-1 (= SHA-160)

Example of ECGDSA over GF(p) with the 192-bit elliptic curve `brainpoolP192r1` and the hash function SHA-1 (= SHA-160)

For the parameters belonging to `brainpoolP192r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-1". The SHA-1 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-1}(m) = \texttt{00000000 CF00CD42 CAA80DDF 8DDEBDFD 32F2DA15 11B53F29}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{22C17C2A 367DD85A B8A365ED 06F19C43 F9ED1834 9A9BC044}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{2D017BE7 F117FF99 4ED6FC63 CA5B4C7A 0430E9FA 095DAFC4}.$$

The value $s = (k \cdot r - \text{SHA-1}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{18FD604E 5F00F55B 3585C052 8C319A2B 05B8F2DD EE9CF1A6}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-1 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-1" is

$$\text{SHA-1}(m) = \texttt{00000000 CF00CD42 CAA80DDF 8DDEBDFD 32F2DA15 11B53F29}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-1}(m) \bmod q$ is

$$u_1 = \texttt{2DCA6B41 0597C195 00DF06B4 C903962D 806CCB36 66DCE213}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{99DDB0F4 D073AE35 0D9B2E17 37D551CA A3C8D87B EA63E40E}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

`2D017BE7 F117FF99 4ED6FC63 CA5B4C7A 0430E9FA 095DAFC4`,

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

2.4.3 Examples of ECGDSA over GF(p) with the hash function SHA-224

Example of ECGDSA over GF(p) with the 256-bit elliptic curve `brainpoolP256r1` and the hash function SHA-224

For the parameters belonging to `brainpoolP256r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224". The SHA-224 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-224}(m) = \texttt{00000000 92AE8A0E 8D08EADE E9426378}$$
$$\texttt{714FF3E0 1957587D 2876FA70 D40E3144}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{908E3099 776261A4 558FF7A9 FA6DFFE0}$$
$$\texttt{CA6BB3F9 CB35C2E4 E1DC73FD 5E8C08A3}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{62CCD1D2 91E62F6A 4FFBD966 C66C85AA}$$
$$\texttt{BA990BB6 AB0C087D BD54A456 CCC84E4C}.$$

The value $s = (k \cdot r - \text{SHA-224}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{6F029D92 1CBD2552 6EDCCF1C 45E3CBF7}$$
$$\texttt{B7A5D8D4 E005F0C4 1C49B052 DECB04EA}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-224 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224" is

$$\text{SHA-224}(m) = \texttt{00000000 92AE8A0E 8D08EADE E9426378}$$
$$\texttt{714FF3E0 1957587D 2876FA70 D40E3144}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-224}(m) \bmod q$ is

$$
\begin{aligned}
u_1 &= \texttt{90BE1C89 4E9F8B01 8511D4CB E53A627D} \\
& \texttt{01A54EA2 0FB6528B 8285885C 704A81A9}.
\end{aligned}
$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$
\begin{aligned}
u_2 &= \texttt{2D22DBCB 6D34B135 DECAA486 3F4BD151} \\
& \texttt{81C1A525 DACD214D E7822365 93E18E76}.
\end{aligned}
$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
62CCD1D2 91E62F6A 4FFBD966 C66C85AA
BA990BB6 AB0C087D BD54A456 CCC84E4C,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 320-bit elliptic curve `brainpoolP320r1` and the hash function SHA-224

For the parameters belonging to `brainpoolP320r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224". The SHA-224 hash value of m (padded with 0's at the high significant end) is

$$\begin{aligned}\text{SHA-224}(m) &= \texttt{00000000 00000000 00000000 92AE8A0E 8D08EADE} \\ & \texttt{E9426378 714FF3E0 1957587D 2876FA70 D40E3144}.\end{aligned}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$\begin{aligned}k &= \texttt{C70BC00A 77AD7872 5D36CEEC 27D6F956 FB546EEF} \\ & \texttt{6DC90E35 31452BD8 7ECE8A4A 7AD730AD C299D81B}.\end{aligned}$$

Then, $r = x(k \cdot G) \bmod q$ is

$$\begin{aligned}r &= \texttt{3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4} \\ & \texttt{FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6}.\end{aligned}$$

The value $s = (k \cdot r - \text{SHA-224}(m)) \cdot d_A \bmod q$ equals

$$\begin{aligned}s &= \texttt{6EA191CA 0D468AC3 E9568768 9338357C 7D0BACB3} \\ & \texttt{F1D87E0D EC05F635 B7ADB842 75AA0086 60F812CF}.\end{aligned}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-224 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224" is

$$\begin{aligned}\text{SHA-224}(m) &= \texttt{00000000 00000000 00000000 92AE8A0E 8D08EADE} \\ & \texttt{E9426378 714FF3E0 1957587D 2876FA70 D40E3144}.\end{aligned}$$

The value $u_1 = r^{-1} \cdot \text{SHA-224}(m) \bmod q$ is

$$\begin{aligned}u_1 &= \texttt{5764B355 FA411E02 293137E3 E40D0E8C 9BC20D7C} \\ & \texttt{C6C36D1D B70E4CD4 6C33C7EB 08FEA888 834C3642}.\end{aligned}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{38075ED1 A3809F2A 29A9BC1D 5D2E4909 7B80232A}$$
$$\texttt{DC7FEDA5 12D6B9E8 F6927B2D AB62DDCD B832FBDB}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4
FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

2.4.4 Examples of ECGDSA over GF(p) with the hash function SHA-256

Example of ECGDSA over GF(p) with the 256-bit elliptic curve `brainpoolP256r1` and the hash function SHA-256

For the parameters belonging to `brainpoolP256r1`, i.e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256". The SHA-256 hash value of m is

$$\text{SHA-256}(m) = \texttt{37ED8AA9 4AE667DB BB753330 E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{908E3099 776261A4 558FF7A9 FA6DFFE0 CA6BB3F9 CB35C2E4 E1DC73FD 5E8C08A3}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{62CCD1D2 91E62F6A 4FFBD966 C66C85AA BA990BB6 AB0C087D BD54A456 CCC84E4C}.$$

The value $s = (k \cdot r - \text{SHA-256}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{1DD53F82 2F8BE769 F601FC58 26B10AB6 03898374 B8501B53 D6976BA1 AAE17A45}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-256 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256" is

$$\text{SHA-256}(m) = \texttt{37ED8AA9 4AE667DB BB753330 E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-256}(m) \bmod q$ is

$$
\begin{aligned}
u_1 &= \texttt{5C9B92A7 D942006F 0D60F667 6DFDD5BA} \\
& \texttt{DBFC447C 78519D4A F7899DD8 26519752}.
\end{aligned}
$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$
\begin{aligned}
u_2 &= \texttt{14527A26 B1376982 656EA211 DA53EE65} \\
& \texttt{3BB619A4 062A7A79 2F4D4F8B D5DED796}.
\end{aligned}
$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
62CCD1D2 91E62F6A 4FFBD966 C66C85AA
BA990BB6 AB0C087D BD54A456 CCC84E4C,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 320-bit elliptic curve `brainpoolP320r1` and the hash function SHA-256

For the parameters belonging to `brainpoolP320r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\text{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256". The SHA-256 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 37ED8AA9 4AE667DB BB753330}$$
$$\texttt{E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{C70BC00A 77AD7872 5D36CEEC 27D6F956 FB546EEF}$$
$$\texttt{6DC90E35 31452BD8 7ECE8A4A 7AD730AD C299D81B}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4}$$
$$\texttt{FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6}.$$

The value $s = (k \cdot r - \text{SHA-256}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{24370797 A9D11717 BBBB2B76 2E08ECD0 7DD7E033}$$
$$\texttt{F544E47C BF3C6D16 FD90B51D CC2E4DD8 E6ECD8CD}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-256 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256" is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 37ED8AA9 4AE667DB BB753330}$$
$$\texttt{E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-256}(m) \bmod q$ is

$$u_1 = \texttt{56E4DCBE 5A4E7C22 5FD218E7 637BAAE9 A5061C64}$$
$$\texttt{115A5947 54BB13E6 2B3EE3CA 200FCD6A 4D2A4C9D}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{AE8E0FDB C3298B2E B0A49532 E353ED07 F1B4D39E 1A44834A 6DEA2213 E3931EB9 E253A473 4C2CEE19}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
3C925969 FAB22F7A E7B8CC5D 50CB0867 DFDB2CF4
FADA3D49 0DF75D72 F7563186 419494C9 8F9C82A6,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 384-bit elliptic curve `brainpoolP384r1` and the hash function SHA-256

384-bit numbers are represented by twelve 32-bit blocks. We write the six high significant 32-bit blocks in the first line and the six low significant 32-bit blocks in the second line.

For `brainpoolP384r1`, the underlying prime p is

$$\begin{aligned} p &= \texttt{8CB91E82 A3386D28 0F5D6F7E 50E641DF 152F7109 ED5456B4} \\ & \texttt{12B1DA19 7FB71123 ACD3A729 901D1A71 87470013 3107EC53} \end{aligned}$$

with 384 bits. The elliptic curve $E: \ y^2 = x^3 + ax + b$ is given by

$$\begin{aligned} a &= \texttt{7BC382C6 3D8C150C 3C72080A CE05AFA0 C2BEA28E 4FB22787} \\ & \texttt{139165EF BA91F90F 8AA5814A 503AD4EB 04A8C7DD 22CE2826} \end{aligned}$$

and

$$\begin{aligned} b &= \texttt{04A8C7DD 22CE2826 8B39B554 16F0447C 2FB77DE1 07DCD2A6} \\ & \texttt{2E880EA5 3EEB62D5 7CB43902 95DBC994 3AB78696 FA504C11}. \end{aligned}$$

Its cardinality is

$$\#E(\mathrm{GF}(p)) \ = \ q,$$

where

$$\begin{aligned} q &= \texttt{8CB91E82 A3386D28 0F5D6F7E 50E641DF 152F7109 ED5456B3} \\ & \texttt{1F166E6C AC0425A7 CF3AB6AF 6B7FC310 3B883202 E9046565} \end{aligned}$$

is a 384-bit prime. $G = (x(G), y(G))$ with

$$\begin{aligned} x(G) &= \texttt{1D1C64F0 68CF45FF A2A63A81 B7C13F6B 8847A3E7 7EF14FE3} \\ & \texttt{DB7FCAFE 0CBD10E8 E826E034 36D646AA EF87B2E2 47D4AF1E} \end{aligned}$$

and

$$\begin{aligned} y(G) &= \texttt{8ABE1D75 20F9C2A4 5CB1EB8E 95CFD552 62B70B29 FEEC5864} \\ & \texttt{E19C054F F9912928 0E464621 77918111 42820341 263C5315} \end{aligned}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$\begin{aligned} d_A &= \texttt{60BABEC4 9D0A4E36 32887959 1B1A598F 339F7971 E8A1AD35} \\ & \texttt{788486EB 081C838B 5612F6DE BD6B38A0 BA720BD8 57AB2354}. \end{aligned}$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{2DE35333 66C51912 4D6D5A05 9313353A A5B5AA35 B7CDC779}$$
$$\texttt{53CEEBF8 7F5FC209 30A62FA3 76877ADB 21117A67 B33CF7C3}$$

and

$$y(P_A) = \texttt{237E4D9E 6A039E85 3A708E38 BF39E94A A587D15C 03BB7F5F}$$
$$\texttt{B1B77EF1 7F67630C 470C0C35 975F6759 B2BB9016 F503A535.}$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256". The SHA-256 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 00000000 00000000 37ED8AA9 4AE667DB}$$
$$\texttt{BB753330 E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7.}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{43E01A2A 95EE7695 95533441 0F32C73B D1394BBF 2CD7B8A1}$$
$$\texttt{8656B447 A951342C 82F52E83 3FFB3B74 61267943 7C13ACB5.}$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{2A2676EF F87A75EE 9ECBA1FD D7A54376 97294166 063C8CD9}$$
$$\texttt{0F8AEBA3 99BF450F FA244C0E E69B3E1F FCA395CD 27AFFC61.}$$

The value $s = (k \cdot r - \text{SHA-256}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{56F6A189 06455867 EB51EBE4 6049A11D 79AEED15 00D1D1A4}$$
$$\texttt{3D876E42 2C9234ED 6F59AB7D 336BCE12 CED3D7EC BC09CAE3.}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-256 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256" is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 00000000 00000000 37ED8AA9 4AE667DB}$$
$$\texttt{BB753330 E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7.}$$

The value $u_1 = r^{-1} \cdot \text{SHA-256}(m) \bmod q$ is

$$u_1 = \texttt{52A28E4B E2F18272 2507ADBB 29B59573 45A1C313 AD56EC8B}$$
$$\texttt{C516B1C2 B8AD8337 1AEE8990 022E5689 264E5B40 91DAA959.}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{73EB284A 7C5ACC66 6DAAB7DF 366792F4 BD06E59C 8465DDC4}$$
$$\texttt{9135A0B6 5058FFB1 4E6B76BF 03DCAA56 BF66C0DB 86FA7ECC}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
2A2676EF F87A75EE 9ECBA1FD D7A54376 97294166 063C8CD9
0F8AEBA3 99BF450F FA244C0E E69B3E1F FCA395CD 27AFFC61,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

2.4.5 Examples of ECGDSA over GF(p) with the hash function SHA-384

Example of ECGDSA over GF(p) with the 384-bit elliptic curve `brainpoolP384r1` and the hash function SHA-384

For the parameters belonging to `brainpoolP384r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.4.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-384". The SHA-384 hash value of m is

$$\text{SHA-384}(m) = \texttt{68FEAB7D 8BF8A779 4466E447 5959946B 2136C084 A86090CA}$$
$$\texttt{8070C980 68B1250D 88213190 6B7E0CB8 475F9054 E9290C2E}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{43E01A2A 95EE7695 95533441 0F32C73B D1394BBF 2CD7B8A1}$$
$$\texttt{8656B447 A951342C 82F52E83 3FFB3B74 61267943 7C13ACB5}.$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{2A2676EF F87A75EE 9ECBA1FD D7A54376 97294166 063C8CD9}$$
$$\texttt{0F8AEBA3 99BF450F FA244C0E E69B3E1F FCA395CD 27AFFC61}.$$

The value $s = (k \cdot r - \text{SHA-384}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{733F4E37 0AF3F9A2 DF9499F9 953E091D 7BD28CA8 E80FB3B4}$$
$$\texttt{AAEB1FF3 24CCDF6E 4D7F6B45 76071321 D8B34C20 CAF0CD01}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-384 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-384" is

$$\text{SHA-384}(m) = \texttt{68FEAB7D 8BF8A779 4466E447 5959946B 2136C084 A86090CA}$$
$$\texttt{8070C980 68B1250D 88213190 6B7E0CB8 475F9054 E9290C2E}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-384}(m) \bmod q$ is

$$
\begin{aligned}
u_1 &= \texttt{5D8FAE65 794BB02E BB44AC1A EC312743 192331BB C42211DB} \\
& \texttt{3350BA11 2A78084F 22B28535 8C10967E 389545F8 0EF883BC}.
\end{aligned}
$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$
\begin{aligned}
u_2 &= \texttt{82B7BB13 30FF3FBA 49CB1224 66D8BCBE D4F9DB7D A722EBB0} \\
& \texttt{F4E693DA 1E758BDE 2D1E3319 8DF2C6AB 706B874D F714A5E0}.
\end{aligned}
$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
2A2676EF F87A75EE 9ECBA1FD D7A54376 97294166 063C8CD9
0F8AEBA3 99BF450F FA244C0E E69B3E1F FCA395CD 27AFFC61,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(p) with the 512-bit elliptic curve `brainpoolP512r1` and the hash function SHA-384

512-bit numbers are represented by sixteen 32-bit blocks. We write four 32-bit blocks in each line, the four most significant 32-bit blocks in the first line and the four least significant 32-bit blocks in the last line.

For `brainpoolP512r1`, the underlying prime p is

$$\begin{array}{rcl} p & = & \texttt{AADD9DB8 DBE9C48B 3FD4E6AE 33C9FC07} \\ & & \texttt{CB308DB3 B3C9D20E D6639CCA 70330871} \\ & & \texttt{7D4D9B00 9BC66842 AECDA12A E6A380E6} \\ & & \texttt{2881FF2F 2D82C685 28AA6056 583A48F3} \end{array}$$

with 512 bits. The elliptic curve $E:\ y^2 = x^3 + ax + b$ is given by

$$\begin{array}{rcl} a & = & \texttt{7830A331 8B603B89 E2327145 AC234CC5} \\ & & \texttt{94CBDD8D 3DF91610 A83441CA EA9863BC} \\ & & \texttt{2DED5D5A A8253AA1 0A2EF1C9 8B9AC8B5} \\ & & \texttt{7F1117A7 2BF2C7B9 E7C1AC4D 77FC94CA} \end{array}$$

and

$$\begin{array}{rcl} b & = & \texttt{3DF91610 A83441CA EA9863BC 2DED5D5A} \\ & & \texttt{A8253AA1 0A2EF1C9 8B9AC8B5 7F1117A7} \\ & & \texttt{2BF2C7B9 E7C1AC4D 77FC94CA DC083E67} \\ & & \texttt{984050B7 5EBAE5DD 2809BD63 8016F723}. \end{array}$$

Its cardinality is

$$\#E(\mathrm{GF}(p)) \ = \ q,$$

where

$$\begin{array}{rcl} q & = & \texttt{AADD9DB8 DBE9C48B 3FD4E6AE 33C9FC07} \\ & & \texttt{CB308DB3 B3C9D20E D6639CCA 70330870} \\ & & \texttt{553E5C41 4CA92619 41866119 7FAC1047} \\ & & \texttt{1DB1D381 085DDADD B5879682 9CA90069} \end{array}$$

is a 512-bit prime. $G = (x(G), y(G))$ with

$$\begin{array}{rcl} x(G) & = & \texttt{81AEE4BD D82ED964 5A21322E 9C4C6A93} \\ & & \texttt{85ED9F70 B5D916C1 B43B62EE F4D0098E} \\ & & \texttt{FF3B1F78 E2D0D48D 50D1687B 93B97D5F} \\ & & \texttt{7C6D5047 406A5E68 8B352209 BCB9F822} \end{array}$$

and

$$
\begin{aligned}
y(G) \;=\; & \texttt{7DDE385D 566332EC C0EABFA9 CF7822FD} \\
& \texttt{F209F700 24A57B1A A000C55B 881F8111} \\
& \texttt{B2DCDE49 4A5F485E 5BCA4BD8 8A2763AE} \\
& \texttt{D1CA2B2F A8F05406 78CD1E0F 3AD80892}
\end{aligned}
$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$
\begin{aligned}
d_A \;=\; & \texttt{92006A98 8AF96D91 57AADCF8 62716962} \\
& \texttt{7CE2ECC4 C58ECE5C 1A0A8642 11AB764C} \\
& \texttt{04236FA0 160857A7 8E71CCAE 4D79D52E} \\
& \texttt{5A69A457 8AF50658 1F598FA9 B4F7DA68}.
\end{aligned}
$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$
\begin{aligned}
x(P_A) \;=\; & \texttt{476784D3 9E2D7B42 AAC3F60F 2DFE3D7C} \\
& \texttt{96278061 2464104B A45C36F3 22F2334C} \\
& \texttt{A5D1FF80 71168925 28104793 4CA9F938} \\
& \texttt{1FD4FD77 F1FB96FC 596DE412 496B95E9}
\end{aligned}
$$

and

$$
\begin{aligned}
y(P_A) \;=\; & \texttt{58AEB51C 58B2D4FF 20636A59 14B63D5E} \\
& \texttt{85B1FA52 20FC968C 1F9AF0EB 64CAA159} \\
& \texttt{30DFD5BB 5BC16B8A F0DBE574 6454BEF9} \\
& \texttt{90DB7F1D 0EDE46E4 655C05B9 032D3E10}.
\end{aligned}
$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-384". The SHA-384 hash value of m (padded with 0's at the high significant end) is

$$
\begin{aligned}
\text{SHA-384}(m) \;=\; & \texttt{00000000 00000000 00000000 00000000} \\
& \texttt{68FEAB7D 8BF8A779 4466E447 5959946B} \\
& \texttt{2136C084 A86090CA 8070C980 68B1250D} \\
& \texttt{88213190 6B7E0CB8 475F9054 E9290C2E}.
\end{aligned}
$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$
\begin{aligned}
k \;=\; & \texttt{6942B01D 5901BEC1 506BB874 9618E22E} \\
& \texttt{C0FCD7F3 5159D51E D53BA77A 78752128} \\
& \texttt{A58232AD 8E0E021A FDE1477F F4C74FDF} \\
& \texttt{FE88AE2D 15D89B56 F6D73C03 77631D2B}.
\end{aligned}
$$

Then, $r = x(k \cdot G) \bmod q$ is

$$r = \texttt{0104918B 2B32B1A5 49BD43C3 0092953B}$$
$$\texttt{4164CA01 A1A97B5B 0756EA06 3AC16B41}$$
$$\texttt{B88A1BAB 4538CD7D 8466180B 3E3F5C86}$$
$$\texttt{46AC4A45 F564E9B6 8FEE72ED 00C7AC48.}$$

The value $s = (k \cdot r - \text{SHA-384}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{3D233E9F D9EB152E 889F4F7C F325B464}$$
$$\texttt{0894E5EA 44C51443 54305CD4 BF70D234}$$
$$\texttt{8257C2DB E06C5544 92CE9FDD 6861A565}$$
$$\texttt{77B53E5E E80E6062 31A4CF06 8FA1EC21.}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-384 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function ␣SHA-384" is

$$\text{SHA-384}(m) = \texttt{00000000 00000000 00000000 00000000}$$
$$\texttt{68FEAB7D 8BF8A779 4466E447 5959946B}$$
$$\texttt{2136C084 A86090CA 8070C980 68B1250D}$$
$$\texttt{88213190 6B7E0CB8 475F9054 E9290C2E.}$$

The value $u_1 = r^{-1} \cdot \text{SHA-384}(m) \bmod q$ is

$$u_1 = \texttt{91A16593 9EE7B5CE 58874782 5C4AD02B}$$
$$\texttt{FFCE6F19 252590FE B5306DCC 193E33F7}$$
$$\texttt{8B25BC4E 3A5AB295 93B418C1 49587165}$$
$$\texttt{E47EBC7F CAB194B3 CD204FA2 E7D2AB70.}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{555596DD 9232BF1E 9D625122 5A9F7C80}$$
$$\texttt{9D5C321C B650F435 C98D61A2 22DC1143}$$
$$\texttt{8327D452 D74193AC 8FBCC6FA F5FFA98E}$$
$$\texttt{8668FCF0 2BE25A85 66FBB268 9CBC02F1.}$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
0104918B 2B32B1A5 49BD43C3 0092953B
4164CA01 A1A97B5B 0756EA06 3AC16B41
B88A1BAB 4538CD7D 8466180B 3E3F5C86
46AC4A45 F564E9B6 8FEE72ED 00C7AC48,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

2.4.6 Example of ECGDSA over GF(p) with the hash function SHA-512

Example of ECGDSA over GF(p) with the 512-bit elliptic curve `brainpoolP512r1` and the hash function SHA-512

For the parameters belonging to `brainpoolP512r1`, i. e., the prime p, the parameters a and b of the elliptic curve E over GF(p), the cardinality q of $E(\mathrm{GF}(p))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, see Section 2.4.5.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-512". The SHA-512 hash value of m is

$$\begin{aligned}\text{SHA-512}(m) \;=\; & \texttt{1A95EF81 D213BD3B 8191E7FE 7F5BFD43} \\ & \texttt{F51E3EE5 A4FD3D08 4A7C9BB5 411F4649} \\ & \texttt{746AEBC6 623D4DEA 7E02DC5A 85E24AF2} \\ & \texttt{96B5A555 AD470413 71E4BF64 380F3E34}.\end{aligned}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$\begin{aligned}k \;=\; & \texttt{6942B01D 5901BEC1 506BB874 9618E22E} \\ & \texttt{C0FCD7F3 5159D51E D53BA77A 78752128} \\ & \texttt{A58232AD 8E0E021A FDE1477F F4C74FDF} \\ & \texttt{FE88AE2D 15D89B56 F6D73C03 77631D2B}.\end{aligned}$$

Then, $r = x(k \cdot G) \bmod q$ is

$$\begin{aligned}r \;=\; & \texttt{0104918B 2B32B1A5 49BD43C3 0092953B} \\ & \texttt{4164CA01 A1A97B5B 0756EA06 3AC16B41} \\ & \texttt{B88A1BAB 4538CD7D 8466180B 3E3F5C86} \\ & \texttt{46AC4A45 F564E9B6 8FEE72ED 00C7AC48}.\end{aligned}$$

The value $s = (k \cdot r - \text{SHA-512}(m)) \cdot d_A \bmod q$ equals

$$\begin{aligned}s \;=\; & \texttt{17A011F8 DD7B5665 2B27AA6D 6E7BDF3C} \\ & \texttt{7C23B5FA 32910FBA A107E627 0E1CA8A7} \\ & \texttt{A263F661 8E6098A0 D6CD6BA1 C03544C5} \\ & \texttt{425875EC B3418AF5 A3EE3F32 143E48D2}.\end{aligned}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-512 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function ␣SHA-512" is

$$\text{SHA-512}(m) = \texttt{1A95EF81 D213BD3B 8191E7FE 7F5BFD43}$$
$$\texttt{F51E3EE5 A4FD3D08 4A7C9BB5 411F4649}$$
$$\texttt{746AEBC6 623D4DEA 7E02DC5A 85E24AF2}$$
$$\texttt{96B5A555 AD470413 71E4BF64 380F3E34}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-512}(m) \bmod q$ is

$$u_1 = \texttt{878E4568 F9A823E8 22350DC2 C83ECA8A}$$
$$\texttt{0CE75958 20B84E3A 5509BF03 2C7E16BD}$$
$$\texttt{ED657F4E C56DB835 E5E9533F B505B12F}$$
$$\texttt{E9EE0260 513081F4 F414C733 F6B9BEC6}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{085A952B 1E1A19A8 96777DA5 0D40AD56}$$
$$\texttt{2FA65EFF E0C7AFC3 5C375459 3E596FC8}$$
$$\texttt{89A6B7B2 3FFA0408 556FE00E 490EFA38}$$
$$\texttt{4A906FD3 31D84A8E FCA998FC 17BC5399}.$$

$x(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
0104918B 2B32B1A5 49BD43C3 0092953B
4164CA01 A1A97B5B 0756EA06 3AC16B41
B88A1BAB 4538CD7D 8466180B 3E3F5C86
46AC4A45 F564E9B6 8FEE72ED 00C7AC48,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

3 The digital signature scheme ECGDSA over $\mathrm{GF}(2^n)$

In this chapter we give a description and provide a reference of the ECGDSA over finite fields with characteristic 2.

3.1 The field $\mathbf{GF}(2^n)$: representation of the generating polynomial and of the field elements

Let n be a positive integer and $f(x)$ an irreducible polynomial of degree n over GF(2), i.e.

$$f(x) \;=\; x^n + f_{n-1}x^{n-1} + \cdots + f_1 x + f_0 \qquad \text{with } f_i \in \{0,1\}.$$

The elements of $\mathrm{GF}(2^n)$ may be represented as polynomials of degree less than n. Addition/multiplication of elements of $\mathrm{GF}(2^n)$ is then addition/multiplication of polynomials over GF(2), plus reduction of the result modulo the irreducible polynomial $f(x)$.

All elements of $\mathrm{GF}(2^n)$, and the generating polynomial $f(x)$ itself, may be written as bit sequence of their coefficients, starting with the coefficient of x^{n-1} and ending with the coefficient of x^0, i. e., in big endian notation, e.g.

$$f \;=\; f_{n-1}\, f_{n-2}\, \ldots\, f_1\, f_0,$$

or in hexadecimal notation, to make the representation shorter. For, e.g., $n = 191$ and $f(x) = x^{191} + x^7 + x^6 + x^4 + 1$, we write in hexadecimal notation, in 32-bit blocks,

$$f \;=\; \texttt{80000000 00000000 00000000 00000000 00000000 000000D1}.$$

3.2 Private and public key

The private key of the signer A is a randomly chosen integer

$$d_A \;\in\; \{1, \ldots, q-1\}.$$

The corresponding public key of A is the point

$$P_A \;=\; (d_A^{-1} \bmod q) \cdot G.$$

3.3 Signature generation

In order to sign the message m with the private key d_A, the signer A performs the following steps:

1) A computes the hash value $h(m)$, $0 \leq h(m) < q$.

2) A chooses a random integer $k \in \{1, \ldots, q-1\}$.

3) A determines

$$r = \pi(k \cdot G) \bmod q.$$

   If $r = 0$, A goes back to step 2) and chooses a new random k.

4) A computes the value

$$s = (k \cdot r - h(m)) \cdot d_A \bmod q.$$

   If $s = 0$, A goes back to step 2) and chooses a new random k.

The pair (r, s) is A's signature of the message m.

3.4 Signature verification

In order to verify whether a pair (r, s) is A's signature of the message m, the verifier B performs the following steps, using A's public key P_A:

1) B checks whether r and s are in $\{1, \ldots, q-1\}$.
   If not, (r, s) is not accepted as A's signature of the message m.

2) B computes the hash value $h(m) < q$.

3) B computes the value

$$u_1 = r^{-1} \cdot h(m) \bmod q.$$

4) B computes the value

$$u_2 = r^{-1} \cdot s \bmod q.$$

5) B determines

$$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q.$$

   If — and only if — this value equals r, i. e., $\pi(u_1 \cdot G + u_2 \cdot P_A) = r \bmod q$, the pair (r, s) is accepted as A's signature of the message m.

3.5 Examples

We write the generating polynomial and all field elements in big endian hexadecimal notation in 32-bit blocks, as described in Section 3.1. And for numbers, we also choose big endian hexadecimal notation in 32-bit blocks, i. e., by

$$a_{8l+7} \dots a_{8l+1} a_{8l} \qquad \dots \qquad a_{23} \dots a_{17} a_{16} \qquad a_{15} \dots a_9 a_8 \qquad a_7 \dots a_1 a_0,$$

where the a_i for $i = 0, \dots, 8l + 7$ are hexadecimal digits, we mean the number

$$a_{8l+7} \cdot 16^{8l+7} \; + \; \dots \; + \; a_1 \cdot 16^1 \; + \; a_0 \cdot 16^0.$$

3.5.1 Examples of ECGDSA over GF(2^n) with the hash function RIPEMD-160

Example of ECGDSA over GF(2^{191}) with the hash function RIPEMD-160

We consider the polynomial $f(x) = x^{191} + x^7 + x^6 + x^4 + 1$, i.e.

$$f = \texttt{80000000 00000000 00000000 00000000 00000000 000000D1},$$

as generating polynomial for GF(2^{191}). The elliptic curve $E: \; y^2 + xy = x^3 + ax^2 + b$ is given by

$$a = 0$$

and

$$b = \texttt{7B4945EE 59A59872 1415BC1F 8F88ACAD D3E9CDD2 91C152A9}.$$

Its cardinality is

$$\#E(\mathrm{GF}(2^{191})) = 4 \cdot q,$$

where

$$q = \texttt{1FFFFFFF FFFFFFFF FFFFFFFF CBA60DA0 4BB074F7 4B6BE7A3}$$

is a 189-bit prime. $G = (x(G), y(G))$ with

$$x(G) = \texttt{72D1DD0E AF00EFFB AB3F4999 047B89B9 C544A975 F9AD28E5}$$

and

$$y(G) = \texttt{4F023F86 B566C855 BC629728 A869FF42 71A5B2EC 7CB01125}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$d_A = \texttt{031DF432 8CF08FC9 A7A7B1F7 A1CC86D0 3926344B 2F1D9DE2}.$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{08496D5C 35021065 D0A7C415 2151CC38 47190E30 BA2C13FF}$$

and

$$y(P_A) = \texttt{41B5A3EF 53FD9B28 2FCE8BC8 AB79BA0E 0E93D4D0 EB45B74E}.$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{19A52ED1 EBA03270 2EDE58CB 44DF3B40 6F69658E 9B56F09C}.$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$r = \texttt{1E96266D 54C8CA12 F7C4D7DF 75648EF5 22BF0843 AFE0E921}.$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{15CC4FB8 3277B0D6 5E4DB710 7350A997 1E6FA2E2 4D7855DD}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{0BF3FEE8 83AA95EF F69F644E DE966CF5 257C8A24 6956A2BD}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{0C681673 8D83B158 FFBA900A EABDE7E5 62A64E6F 067339C8}.$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

$$\texttt{1E96266D 54C8CA12 F7C4D7DF 75648EF5 22BF0843 AFE0E921},$$

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(2^{251}) with the hash function RIPEMD-160

251 bits are represented by eight 32-bit blocks. We write the four high significant 32-bit blocks in the first line and the four low significant 32-bit blocks in the second line.

We consider the polynomial $f(x) = x^{251} + x^7 + x^4 + x^2 + 1$, i.e.

$$
\begin{aligned}
f &= \texttt{08000000 00000000 00000000 00000000} \\
  & \texttt{00000000 00000000 00000000 00000095},
\end{aligned}
$$

as generating polynomial for GF(2^{251}). The elliptic curve $E:\ y^2 + xy = x^3 + ax^2 + b$ is given by

$$a = 0$$

and

$$
\begin{aligned}
b &= \texttt{03E0554C 3A8E9D76 268EB11F EB17B6E4} \\
  & \texttt{71755380 253C3B62 BCF6DF05 458F6841}.
\end{aligned}
$$

Its cardinality is

$$\#E(\mathrm{GF}(2^{251})) = 4 \cdot q,$$

where

$$
\begin{aligned}
q &= \texttt{01FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\
  & \texttt{EECAC7DB 054337E4 1CFBF4A3 82A1B15B}
\end{aligned}
$$

is a 249-bit prime. $G = (x(G), y(G))$ with

$$
\begin{aligned}
x(G) &= \texttt{04E4420F 978D4546 E21064F2 1EA3E1CB} \\
  & \texttt{24F0B047 37758609 D27AB383 646713BB}
\end{aligned}
$$

and

$$
\begin{aligned}
y(G) &= \texttt{044C434E D7100229 C617E967 8D003932} \\
  & \texttt{ECF03A71 5A0356FC FD504C80 E3256D89}
\end{aligned}
$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$
\begin{aligned}
d_A &= \texttt{01B9ECE1 1FE404D3 CC657BF0 6B75DFCA} \\
  & \texttt{E6B8F9B5 7E5FAB41 D397816B 68FAC1E0}.
\end{aligned}
$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{071F7C73 AC8DCA9E 8100E6F7 CFC73AA7}$$
$$\texttt{7C226C90 D859B562 BFE894BD 9923EDA5}$$

and

$$y(P_A) = \texttt{03BE97FB 04F7D01F BD76F7BD B12DACEC}$$
$$\texttt{D0687AA1 BC2A5A7F 84166C15 D0FFCC5D}.$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 577EF842}$$
$$\texttt{B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{0197101E DC80C6C1 688C276E E68B2721}$$
$$\texttt{7236F609 AC7283FE 621622E4 A3307332}.$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$r = \texttt{01EC8435 28F70855 D85C6F73 74B94D0E}$$
$$\texttt{D45E14BC 4C067739 3A5E5C2C E98AA549}.$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{0033CD7D BAA5B6ED 1239B9F9 55772447}$$
$$\texttt{B68D0C53 CC82206B 8F72AC51 D7E15B54}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 577EF842}$$
$$\texttt{B32FDE45 79727FFF 02F7A280 74ADC4EF}.$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{01B4F0A2 339B4C44 8F562CC4 88F394F8}$$
$$\texttt{A8AA811D 876DE53F B32F2096 501C1C33}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{0158231A 5BFE8A12 07582308 0AA890B1}$$
$$\texttt{2494D7CD 31F779E4 45E20F6B 1ABE0E20}.$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
01EC8435 28F70855 D85C6F73 74B94D0E
D45E14BC 4C067739 3A5E5C2C E98AA549,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over GF(2^{317}) with the hash function RIPEMD-160

317 bits are represented by ten 32-bit blocks. We write the five high significant 32-bit blocks in the first line and the five low significant 32-bit blocks in the second line.

We consider the polynomial $f(x) = x^{317} + x^{21} + x^9 + x^2 + 1$, i.e.

$$\begin{aligned} f \;=\; & \texttt{20000000 00000000 00000000 00000000 00000000} \\ & \texttt{00000000 00000000 00000000 00000000 00200205}, \end{aligned}$$

as generating polynomial for GF(2^{317}). The elliptic curve $E:\ y^2 + xy = x^3 + ax^2 + b$ is given by

$$a \;=\; 0$$

and

$$\begin{aligned} b \;=\; & \texttt{0770135D F5BA3FAE 6D667223 44009825 7D5D3F6E} \\ & \texttt{503B8B6C 7EB14FAF 24987EB2 07C7EE4E 20C8D0A5}. \end{aligned}$$

Its cardinality is

$$\#E(\mathrm{GF}(2^{317})) \;=\; 4 \cdot q,$$

where

$$\begin{aligned} q \;=\; & \texttt{07FFFFFF FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF} \\ & \texttt{F289C24C 100CC91E 9D92C7AE 16344D63 9AE16FDD} \end{aligned}$$

is a 315-bit prime. $G = (x(G), y(G))$ with

$$\begin{aligned} x(G) \;=\; & \texttt{1BFFA3C3 E5A959ED A89EC946 E19919A8 DE0637CE} \\ & \texttt{2962E972 DD4B8558 47F44D79 64E7EAB3 193FB545} \end{aligned}$$

and

$$\begin{aligned} y(G) \;=\; & \texttt{01CCBA19 375434EE 6805E8A5 F783A472 D6881D28} \\ & \texttt{9FD9969E AE4C9CC9 1945D22B AB3671D9 FEF42ED8} \end{aligned}$$

is a point of order q on E.

signer A's private and public key:

As private key, the signer A chooses the following random integer $d_A \in \{1, \ldots, q-1\}$:

$$\begin{aligned} d_A \;=\; & \texttt{02E60C98 83379190 C8CF0EE2 CF037C66 EA4C5F02} \\ & \texttt{84BF79B4 6F84A31D D6FCBF84 3EDD71E7 5E7A8413}. \end{aligned}$$

A's public key is then the point $P_A = (d_A^{-1} \bmod q) \cdot G$ with the coordinates

$$x(P_A) = \texttt{09012A89 9CFFBDA0 76476517 51CD4345 906E9444}$$
$$\texttt{A7A293A2 61D741C0 DF342CDE 18351403 3B62DE6F}$$

and

$$y(P_A) = \texttt{047EC184 856ECA0E D6A8A55A F3637529 49494A0C}$$
$$\texttt{49CCF165 38450BA7 FDFDBA36 B8CB9FF7 4FCCACEB.}$$

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160". The RIPEMD-160 hash value of m (padded with 0's at the high significant end) is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 00000000 00000000}$$
$$\texttt{577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF.}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{006A7A58 7E54463A 31B248EE 01E0828F 230DD6DB}$$
$$\texttt{B7D91A6D 8696887C 5AC5038C F2BB7192 59205F14.}$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$r = \texttt{02B8E635 81F0B94A 641A040F 669BD05A 5FABD963}$$
$$\texttt{5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A.}$$

The value $s = (k \cdot r - \text{RIPEMD-160}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{054024D1 3EFC89BF C4752B90 60EAC47A 67BAAA50}$$
$$\texttt{6B1FAD27 6C5FB152 31802DEC 21BAD505 B7566812.}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The RIPEMD-160 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣RIPEMD-160" is

$$\text{RIPEMD-160}(m) = \texttt{00000000 00000000 00000000 00000000 00000000}$$
$$\texttt{577EF842 B32FDE45 79727FFF 02F7A280 74ADC4EF.}$$

The value $u_1 = r^{-1} \cdot \text{RIPEMD-160}(m) \bmod q$ is

$$u_1 = \texttt{00ED9526 90A54B97 4D7609D1 38A23CFF 897C07CD}$$
$$\texttt{DDBEA003 C86383BA F06CCD70 23E18B5C 7C575DFC.}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{0129A59B 3E7F77D8 65BE93FC 10AC5241 1E8C2CC7}$$
$$\texttt{35CDA4E1 AF40D195 93ED2885 295D96CB 53840B10}.$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
02B8E635 81F0B94A 641A040F 669BD05A 5FABD963
5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

3.5.2 Examples of ECGDSA over GF(2^n) with the hash function SHA-1 (=SHA-160)

Example of ECGDSA over GF(2^{191}) with the hash function SHA-1 (=SHA-160)

We consider the same elliptic curve as in Section 3.5.1; for the generating polynomial f of GF(2^n), the parameters a and b of the elliptic curve E over GF(2^n), the prime q such that $4 \cdot q$ is the cardinality of $E(\mathrm{GF}(2^n))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, please refer to Section 3.5.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-1". The SHA-1 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-1}(m) = \texttt{00000000 CF00CD42 CAA80DDF 8DDEBDFD 32F2DA15 11B53F29.}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$k = \texttt{19A52ED1 EBA03270 2EDE58CB 44DF3B40 6F69658E 9B56F09C.}$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$r = \texttt{1E96266D 54C8CA12 F7C4D7DF 75648EF5 22BF0843 AFE0E921.}$$

The value $s = (k \cdot r - \text{SHA-1}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{172E6339 A6BA069A 43E252F1 D4F8F407 051D22D3 AB4F3B65.}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-1 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-1" is

$$\text{SHA-1}(m) = \texttt{00000000 CF00CD42 CAA80DDF 8DDEBDFD 32F2DA15 11B53F29.}$$

The value $u_1 = r^{-1} \cdot \text{SHA-1}(m) \bmod q$ is

$$u_1 = \texttt{1A0CFA32 AD36D067 34CA3149 00068D0A 5F7B19AE 14225185.}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{065913B1 29A632D6 BEE7BF97 C8DECE9A 51B9822A B5162C1D.}$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

`1E96266D 54C8CA12 F7C4D7DF 75648EF5 22BF0843 AFE0E921`,

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

3.5.3 Examples of ECGDSA over $\mathrm{GF}(2^{\mathrm{n}})$ with the hash function SHA-224

Example of ECGDSA over $\mathrm{GF}(2^{251})$ with the hash function SHA-224

We consider the same elliptic curve as in Section 3.5.1; for the generating polynomial f of $\mathrm{GF}(2^n)$, the parameters a and b of the elliptic curve E over $\mathrm{GF}(2^n)$, the prime q such that $4 \cdot q$ is the cardinality of $E(\mathrm{GF}(2^n))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, please refer to Section 3.5.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224". The SHA-224 hash value of m (padded with 0's at the high significant end) is

$$\begin{aligned} \text{SHA-224}(m) &= \texttt{00000000 92AE8A0E 8D08EADE E9426378} \\ & \texttt{714FF3E0 1957587D 2876FA70 D40E3144}. \end{aligned}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$\begin{aligned} k &= \texttt{0197101E DC80C6C1 688C276E E68B2721} \\ & \texttt{7236F609 AC7283FE 621622E4 A3307332}. \end{aligned}$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$\begin{aligned} r &= \texttt{01EC8435 28F70855 D85C6F73 74B94D0E} \\ & \texttt{D45E14BC 4C067739 3A5E5C2C E98AA549}. \end{aligned}$$

The value $s = (k \cdot r - \text{SHA-224}(m)) \cdot d_A \bmod q$ equals

$$\begin{aligned} s &= \texttt{01858FDF 9BB59678 5327C98D 66595959} \\ & \texttt{9CF3390A FF948476 7D605953 C9809D38}. \end{aligned}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-224 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224" is

$$\begin{aligned} \text{SHA-224}(m) &= \texttt{00000000 92AE8A0E 8D08EADE E9426378} \\ & \texttt{714FF3E0 1957587D 2876FA70 D40E3144}. \end{aligned}$$

The value $u_1 = r^{-1} \cdot \text{SHA-224}(m) \bmod q$ is

$$
\begin{aligned}
u_1 &= \texttt{00BACB27 DF7A1062 2D574BA4 7E994596} \\
& \texttt{FFDA209E 064E8D8F 3613E378 B427AEBC}.
\end{aligned}
$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$
\begin{aligned}
u_2 &= \texttt{000342A9 F9129C30 40A69D8D 11705B6E} \\
& \texttt{C0D9BE18 9C05FCF1 89713F60 239840B6}.
\end{aligned}
$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
01EC8435 28F70855 D85C6F73 74B94D0E
D45E14BC 4C067739 3A5E5C2C E98AA549,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

Example of ECGDSA over $GF(2^{317})$ with the hash function SHA-224

We consider the same elliptic curve as in Section 3.5.1; for the generating polynomial f of $GF(2^n)$, the parameters a and b of the elliptic curve E over $GF(2^n)$, the prime q such that $4 \cdot q$ is the cardinality of $E(GF(2^n))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, please refer to Section 3.5.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224". The SHA-224 hash value of m (padded with 0's at the high significant end) is

$$\begin{aligned}\text{SHA-224}(m) &= \texttt{00000000 00000000 00000000 92AE8A0E 8D08EADE} \\ & \texttt{E9426378 714FF3E0 1957587D 2876FA70 D40E3144}.\end{aligned}$$

The signer A chooses the following random integer $k \in \{1, \ldots, q-1\}$:

$$\begin{aligned}k &= \texttt{006A7A58 7E54463A 31B248EE 01E0828F 230DD6DB} \\ & \texttt{B7D91A6D 8696887C 5AC5038C F2BB7192 59205F14}.\end{aligned}$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$\begin{aligned}r &= \texttt{02B8E635 81F0B94A 641A040F 669BD05A 5FABD963} \\ & \texttt{5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A}.\end{aligned}$$

The value $s = (k \cdot r - \text{SHA-224}(m)) \cdot d_A \bmod q$ equals

$$\begin{aligned}s &= \texttt{0011499E 9493974B 0D711BFE 05118AAB D05DCDE5} \\ & \texttt{E70DCB65 7BBBF8AC C5B2DBD3 21FAE2AF F75932F3}.\end{aligned}$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \ldots, q-1\}$.

The SHA-224 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-224" is

$$\begin{aligned}\text{SHA-224}(m) &= \texttt{00000000 00000000 00000000 92AE8A0E 8D08EADE} \\ & \texttt{E9426378 714FF3E0 1957587D 2876FA70 D40E3144}.\end{aligned}$$

The value $u_1 = r^{-1} \cdot \text{SHA-224}(m) \bmod q$ is

$$\begin{aligned}u_1 &= \texttt{0298F462 D8A96014 E1DED8A9 783AD2BD 7B28C3B2} \\ & \texttt{B43E367F E1B08491 0178CF5A 1A61FE21 6DBEB4C5}.\end{aligned}$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{0396DB8D 40972DD5 EA8B2373 12FD5FAE 8F205C8E}$$
$$\texttt{89320FFB C64C2761 1C05C88C 5FC2E29C 0A346708}.$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
02B8E635 81F0B94A 641A040F 669BD05A 5FABD963
5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

3.5.4 Example of ECGDSA over GF(2^n) with the hash function SHA-256

Example of ECGDSA over GF(2^{317}) with the hash function SHA-256

We consider the same elliptic curve as in Section 3.5.1; for the generating polynomial f of GF(2^n), the parameters a and b of the elliptic curve E over GF(2^n), the prime q such that $4 \cdot q$ is the cardinality of $E(\mathrm{GF}(2^n))$, the coordinates $x(G)$ and $y(G)$ of the point G of order q on E, the private key d_A, and the public key P_A, please refer to Section 3.5.1.

signature generation:

Let m be the ASCII coded message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256". The SHA-256 hash value of m (padded with 0's at the high significant end) is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 37ED8AA9 4AE667DB BB753330}$$
$$\texttt{E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The signer A chooses the following random integer $k \in \{1, \dots, q-1\}$:

$$k = \texttt{006A7A58 7E54463A 31B248EE 01E0828F 230DD6DB}$$
$$\texttt{B7D91A6D 8696887C 5AC5038C F2BB7192 59205F14}.$$

Then, $r = \pi(k \cdot G) \bmod q$ is

$$r = \texttt{02B8E635 81F0B94A 641A040F 669BD05A 5FABD963}$$
$$\texttt{5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A}.$$

The value $s = (k \cdot r - \text{SHA-256}(m)) \cdot d_A \bmod q$ equals

$$s = \texttt{02C2839A 6DE7BB13 DE343A92 216EFE64 289CF506}$$
$$\texttt{D7532ECE 4DE57D4E 5064B2AE 39BB1F0F CFE4815E}.$$

The pair (r, s) is A's signature of the message m.

signature verification:

r and s as above lie in $\{1, \dots, q-1\}$.

The SHA-256 hash value of the message "Example␣of␣ECGDSA␣with␣the␣hash␣function␣SHA-256" is

$$\text{SHA-256}(m) = \texttt{00000000 00000000 37ED8AA9 4AE667DB BB753330}$$
$$\texttt{E050EB8E 12195807 ECDC4FB1 0E0662B4 22C219D7}.$$

The value $u_1 = r^{-1} \cdot \text{SHA-256}(m) \bmod q$ is

$$u_1 = \texttt{03B3C8B2 A8A0FF67 1FCAE053 5167EAD4 7860744D}$$
$$\texttt{3E97A16F BD52BF5E 18BF11A5 A0C3014A 434DC14F}.$$

$u_2 = r^{-1} \cdot s \bmod q$ equals

$$u_2 = \texttt{038BCED3 DF98B7E1 BFFCD2A4 92F2B290 8E34FB6A}$$
$$\texttt{2B50C9CA 6F0C8C19 ACDAA417 A282955A 3F074A10}.$$

$\pi(u_1 \cdot G + u_2 \cdot P_A) \bmod q$ is

```
02B8E635 81F0B94A 641A040F 669BD05A 5FABD963
5AECC0B6 3AFA6848 F290F72A 868E7B80 5976035A,
```

which is equal to r, so that the pair (r, s) is accepted as A's signature of the message m.

4 ASN.1 Syntax Specification for the digital signature scheme ECGDSA

In this chapter the syntax for elliptic curve domain parameters, signatures and keys according to Abstract Syntax Notation One (ASN.1) is given. If elliptic curve domain parameters, signatures and keys are represented in ASN.1 syntax, then their syntax should be as defined here.

The ASN.1 syntax specified in the following sections follows widely the syntax specified for elliptic curve based cryptographic schemes in ANSI X9.62 [6], [8] and ANSI X9.63 [7].

This specification makes use of several object identifiers defined in [7]. The object identifier `ansi-X9-62` represents the root of the tree containing all object identifiers defined in [6], and has the following value:

```
ansi-X9-62 OBJECT IDENTIFIER ::= {iso(1) member-body(2) us(840)
                                  10045}
```

The object identifier `ecgDsaStd` represents the root of the tree containing all object identifiers additionally defined in this document, and has the following value:

```
ecgDsaStd OBJECT IDENTIFIER ::= {iso(1) identified-organization(3)
                                 teletrust(36) algorithm(3)
                               signature-algorithm(3) ecSign(2) ecgDsa(5)}
```

4.1 Syntax for Finite Fields

This section provides the syntax for the finite fields defined in this document.

A finite field shall be defined by a value of the parameterized type `FieldID`:

```
FieldID{FIELD-ID:IOSet} ::= SEQUENCE {
 fieldType   FIELD-ID.&id({IOSet}),
 parameters  FIELD-ID.&Type({IOSet}{@fieldType})
}
```

The type `FieldID` is composed of two components, `fieldType` and `parameters`, which are specified by the fields `&id` and `&Type`. Both fields form a template for defining sets of information objects, instances of the class `FIELD-ID`.

```
FIELD-ID ::= TYPE-IDENTIFIER
```

The class `FIELD-ID` is based on the information object class `TYPE-IDENTIFIER`, which is described in [9], Annex A.

In an instance of `FieldID`, the component `fieldType` will contain an object identifier value that uniquely identifies the type contained in component `parameters`. The component relation constraint `IOSet@fieldType` binds the argument `IOSet` to the object identifier `fieldType`.

In this document, two finite field types are permitted: prime fields and characteristic-two fields. The object identifier `id-fieldType` represents the root of a tree containing the object identifiers of each finite field type:

```
id-fieldType OBJECT IDENTIFIER ::= {ansi-X9-62 fieldType(1)}
```

The information object set `FieldTypes` contains two objects, where each object represents a finite field type:

```
FieldTypes FIELD-ID ::= { {Prime-p IDENTIFIED BY prime-field} |
{Characteristic2Set IDENTIFIED BY characteristic-two-field},
 ...
}
```

Each object of the object set `FieldTypes` contains a unique object identifier and an associated type. Object identifier and associated type for the different finite field types are described in the next sections.

Syntax for Prime Fields

The finite field GF(p), where p is an odd prime, is specified by the object identifier `prime-field`:

```
prime-field OBJECT IDENTIFIER ::= {id-fieldType 1}
```

This object identifier is associated with type `Prime-p`, which specifies the prime number p:

```
Prime-p ::= INTEGER  -- odd prime
```

Syntax for characteristic-two Fields

The characteristic-two finite field GF(2^m) is specified by the object identifier `characteristic-two-field`:

```
characteristic-two-field OBJECT IDENTIFIER ::= {id-fieldType 2}
```

This object identifier is associated with type set `Characteristic2Set`:

```
Characteristic2Set ::= Characteristic-two {{F2mBasisTypes}}

Characteristic-two {CHARACTERISTIC-TWO:IOSet} ::= SEQUENCE {
 m           INTEGER
 basis       CHARACTERISTIC-TWO.&id({F2mBasisTypes}),
 parameters  CHARACTERISTIC-TWO.&Type({F2mBasisTypes}{@basis})
}

CHARACTERISTIC-TWO ::= TYPE-IDENTIFIER
```

The components of type `Characteristic-two{}` have the following meaning:

- The component `m` specifies the degree of the finite field.
- The component `basis` refers to the basis type used to express the field elements.
- The component `parameters` specifies the reduction polynomial used to generate the field.

The information object set `F2mBasisTypes` is used as the single parameter in a reference to type `Characteristic-two{}`. The set `F2mBasisTypes` holds four objects, where each object describes a single characteristic-two base type valid in this document. The values of these objects define all of the valid values that may appear in an instance of `Characteristic-two{}`.

Each object of the object set `F2mBasisTypes` contains a unique object identifier and an associated parameter type:

```
F2mBasisTypes CHARACTERISTIC-TWO ::= {
 {NULL           IDENTIFIED BY gnBasis} |
 {Trinomial      IDENTIFIED BY tpBasis} |
 {Pentanomial    IDENTIFIED BY ppBasis} |
 {IrrPolynomial  IDENTIFIED BY ipBasis},
 ...
}
```

The syntax above specifies that normal bases, trinomial bases and pentanomial bases are of interest in this document. In contrast to [6] and [7], a base generated by an arbitrary irreducible reduction polynomial is specified as well.

The object identifier `id-characteristic-two-basis` represents the root of a tree containing the object identifiers for each type of basis for the characteristic-two finite fields. It has the following value:

```
id-characteristic-two-basis OBJECT IDENTIFIER ::= {
 characteristic-two-field basisType(3)}
```

Normal bases are specified by the object identifier `gnBasis` with `NULL` parameters.

```
gnBasis OBJECT IDENTIFIER ::= {id-characteristic-two-basis 1}
```

Trinomial bases are specified by the object identifier `tpBasis` with a parameter `Trinomial` describing the integer k where $x^m + x^k + 1$ is the reduction polynomial.

```
tpBasis OBJECT IDENTIFIER ::= {id-characteristic-two-basis 2}
```

```
Trinomial ::= INTEGER
```

Pentanomial bases are specified by the object identifier `ppBasis` with a parameter `Pentanomial`:

```
ppBasis OBJECT IDENTIFIER ::= {id-characteristic-two-basis 3}

Pentanomial ::= SEQUENCE {
 k1  INTEGER,    -- k1 > 0,
 k2  INTEGER,    -- k2 > k1
 k3  INTEGER     -- k3 > k2
}
```

The components `k1`, `k2`, and `k3` of `Pentanomial` specify the integers k_1, k_2, and k_3 where $x^m + x^{k_3} + x^{k_2} + x^{k_1} + 1$ is the reduction polynomial.

Polynomial bases different from trinomial or pentanomial bases are specified by the new object identifier `ipBasis` with a parameter `IrrPolynomial` describing the irreducible reduction polynomial:

```
ecgFieldType OBJECT IDENTIFIER ::= {ecgDsaStd fieldType(1)}

characteristicTwoField OBJECT IDENTIFIER ::= {ecgFieldType 1}

characteristicTwoBasis  OBJECT IDENTIFIER ::=
 {characteristicTwoField basisType(1)}

ipBasis OBJECT IDENTIFIER ::= {characteristicTwoBasis 1}

IrrPolynomial ::= FieldElement
```

For an irreducible polynomial $f(x) = x^m + 1 + \sum_{i=1}^{m-1} a_i x^i$ of degree m generating GF(2^m), there is no representation as field element of GF(2^m). In order to obtain a field element representation of $f(x)$ in parameter `IrrPolynomial`, the field element (in polynomial representation) $f(x) \bmod x^m = 1 + \sum_{i=1}^{m-1} a_i x^i$ is used instead. A description of the type `FieldElement` is given in the next section.

4.2 Syntax for Finite Field Elements and Elliptic Curve Points

This section provides the syntax for the finite field elements and elliptic curve points defined in this document.

A finite field element shall be represented by a value of type `FieldElement`:

```
FieldElement ::= OCTET STRING
```

The value of `FieldElement` shall be the octet string representation of a finite field element following the conversion routine in [6], Section 4.3.3.

An elliptic curve point shall be represented by a value of type `ECPoint`:

```
ECPoint ::= OCTET STRING
```

The value of `ECPoint` shall be the octet string representation of an elliptic curve point following the conversion routine in [6], Section 4.3.6.

4.3 Syntax for Elliptic Curve Domain Parameters

This section provides the syntax for the elliptic curve domain parameters defined in this document.

Elliptic curve domain parameters shall be represented by a value of type `ECParameters`:

```
ECParameters ::= SEQUENCE {
 version  INTEGER{ecpVer1(1)}(ecpVer1),
 fieldID  FieldID{{FieldTypes}},
 curve    Curve,
 base     ECPoint,
 order    INTEGER,
 cofactor INTEGER OPTIONAL
 ...
}
```

The components have the following meaning:

- The component `version` specifies the version number of the elliptic curve domain parameters. In the syntax above, the `INTEGER` element `ecpVer1` is created, whose value is set to 1.
- The component `fieldID` of type `FieldID{}` refers to the finite field over which the elliptic curve is defined. The parameterized type `FieldID{}` was introduced in Section 4.1.
- The component `curve` of type `Curve` specifies the elliptic curve. This type is defined below.
- The component `base` of type `ECPoint` specifies the base point on the elliptic curve.
- The component `order` of type `INTEGER` specifies the order n of the base point.
- The optional component `cofactor` of type `INTEGER` is the integer $h = \sharp E(\mathrm{GF}(q))/n$.

The type `Curve` specifies the coefficients `a` and `b` of the elliptic curve in question:

```
Curve ::= SEQUENCE {
 a     FieldElement,
 b     FieldElement,
 seed  BIT STRING OPTIONAL
}
```

Each coefficient shall be represented as a value of type `FieldElement`. The optional component `seed` may be used to derive the coefficients of a randomly generated elliptic curve.

4.4 Syntax for Public Keys

This section provides the syntax for the public keys defined in this document.

An elliptic curve public key may be represented as value of the X.509 type `SubjectPublicKeyInfo`. In this case the public key shall have the following syntax:

```
SubjectPublicKeyInfo ::= SEQUENCE {
 algorithm          AlgorithmIdentifier{{ECGPKAlgorithms}},
 subjectPublicKey   BIT STRING
}
```

The component `algorithm` of type `AlgorithmIdentifier{}` specifies the type of public key and its associated parameter. The component `subjectPublicKey` of type `BIT STRING` specifies the value of the public key. The elliptic curve public key is a value of type `ECPoint`, which is simply an `OCTET STRING`. The mapping from a value of type `OCTET STRING` to a value of type `BIT STRING` is as follows: the most significant bit of the `OCTET STRING` value becomes the most significant bit of the `BIT STRING` value, etc. And the least significant bit of the `OCTET STRING` becomes the least significant bit of the `BIT STRING`.

The parameter type `AlgorithmIdentifier{}` binds together a set of algorithm object identifiers and their associated parameter types. The type `AlgorithmIdentifier{}` is defined as follows:

```
ALGORITHM ::= CLASS {
 &id OBJECT IDENTIFIER UNIQUE,
 &Type OPTIONAL
} WITH SYNTAX { OID &id [PARMS &Type] }

AlgorithmIdentifier{ALGORITHM:IOSet} ::= SEQUENCE {
 algorithm   ALGORITHM.&id({IOSet}),
 parameters  ALGORITHM.&Type({IOSet}{@algorithm}) OPTIONAL
}
```

The single parameter `ECGPKAlgorithms` in the reference of type `AlgorithmIdentifier{}` specifies all pairs of valid values of this type. This information object set of class `ALGORITHM` contains the seven objects `ecgPublicKeyType`, `ecgdsa-RIPEMD160`, `ecgdsa-SHA1`, `ecgdsa-SHA224`, `ecgdsa-SHA256`, `ecgdsa-SHA384`, and `ecgdsa-SHA512`:

```
ECGPKAlgorithms ALGORITHM ::= {
 ecgPublicKeyType | ecgdsa-RIPEMD160 |
 ecgdsa-SHA1      | ecgdsa-SHA224    |
 ecgdsa-SHA256    | ecgdsa-SHA384    |
 ecgdsa-SHA512,
 ...
}

ecgPublicKeyType ALGORITHM ::= {
 OID ecgPublicKey PARMS Parameters {{ExtendedCurveNames}}
```

```
}

ecgdsa-RIPEMD160 ALGORITHM ::= {
 OID ecgSignatureWithripemd160 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA1 ALGORITHM ::= {
 OID ecgSignatureWithsha1 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA224 ALGORITHM ::= {
 OID ecgSignatureWithsha224 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA256 ALGORITHM ::= {
 OID ecgSignatureWithsha256 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA384 ALGORITHM ::= {
 OID ecgSignatureWithsha384 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA512 ALGORITHM ::= {
 OID ecgSignatureWithsha512 PARMS Parameters {{ExtendedCurveNames}}
}
```

When using one of the object identifiers, the inclusion of the parameters is mandatory.

The object identifier `ecgPublicKey` denotes the public key type defined in this document. It has the following value:

```
ecgKeyType OBJECT IDENTIFIER ::= {ecgDsaStd keyType(2)}

ecgPublicKey OBJECT IDENTIFIER ::= {ecgKeyType publicKey(1)}
```

The object identifiers `ecgSignatureWithripemd160`, `ecgSignatureWithsha1`, `ecgSignatureWithsha224`, `ecgSignatureWithsha256`, `ecgSignatureWithsha384`, and `ecgSignatureWithsha512` are defined in Section 4.6.

The type `Parameters`, a choice of three alternatives, is used to convey parameters associated with elliptic curve public keys:

```
Parameters{CURVES:IOSet} ::= CHOICE {
 ecParameters ECParameters,
 namedCurve CURVES.&id({IOSet}),
 implicitlyCA NULL
}
```

The meaning of the choices is as follows:

- The choice `ecParameters` of type `ECParameters` allows detailed specification of all required values of the elliptic curve domain parameters. The type `ECParameters` was described in Section 4.3.

- The choice `namedCurve` indicates the elliptic curve domain parameters by a reference to a known set of parameters.

- The choice `implicitlyCA` indicates that the elliptic curve domain parameters are explicitly defined elsewhere.

The valid values for the `namedCurve` choice are constrained to those within the class `CURVES`.

```
CURVES ::= CLASS{
 &id  OBJECT IDENTIFIER UNIQUE
} WITH SYNTAX {ID &id}
```

The information object set `ExtendedCurveNames` provides allowed values of the `namedCurve` choice for the mandatory parameters of the information object set `ECGPKAlgorithms`.

```
ExtendedCurveNames CURVES ::= {
 {ANSICurveNames} |
 {BrainpoolCurveNames},
 ...
}
```

The definition of the object set `ANSICurveNames` can be found in [7]. In this document, the usage of the example elliptic curve domain parameters given in [3] is also specified:

```
BrainpoolCurveNames CURVES ::= {
 {ID brainpoolP160r1} |
 {ID brainpoolP160t1} |
 {ID brainpoolP192r1} |
 {ID brainpoolP192t1} |
 {ID brainpoolP224r1} |
 {ID brainpoolP224t1} |
 {ID brainpoolP256r1} |
 {ID brainpoolP256t1} |
 {ID brainpoolP320r1} |
 {ID brainpoolP320t1} |
 {ID brainpoolP384r1} |
 {ID brainpoolP384t1} |
 {ID brainpoolP512r1} |
 {ID brainpoolP512t1},
 ...
}
```

Each curve object is represented as a unique object identifier value. The curve with curve identifier name `brainpoolPLrj` is the jth curve provided by the ECC-Brainpool over the finite field GF(p), where p is an L-bit prime and the coefficient a is selected randomly according to the procedures described in [3], Section 5. The curve with curve identifier name `brainpoolPLrj` is GF(p)-isomorphic to the twisted curve with curve name `brainpoolPLtj` with coefficient $a = -3 \bmod p$. The object identifier `versionOne` represents the tree containing the object identifiers for each set of elliptic curve domain parameters as specified in [3]. The object identifier has the following value:

```
ecc-brainpool OBJECT IDENTIFIER ::= {
 iso(1) identified-organization(3) teletrust(36) algorithm(3)
  signature-algorithm(3) ecSign(2) ecStdCurvesAndGeneration(8)}

ellipticCurve OBJECT IDENTIFIER ::= {ecc-brainpool 1}

versionOne OBJECT IDENTIFIER ::= {ellipticCurve 1}

brainpoolP160r1 OBJECT IDENTIFIER ::= {versionOne 1}
brainpoolP160t1 OBJECT IDENTIFIER ::= {versionOne 2}
brainpoolP192r1 OBJECT IDENTIFIER ::= {versionOne 3}
brainpoolP192t1 OBJECT IDENTIFIER ::= {versionOne 4}
brainpoolP224r1 OBJECT IDENTIFIER ::= {versionOne 5}
brainpoolP224t1 OBJECT IDENTIFIER ::= {versionOne 6}
brainpoolP256r1 OBJECT IDENTIFIER ::= {versionOne 7}
brainpoolP256t1 OBJECT IDENTIFIER ::= {versionOne 8}
brainpoolP320r1 OBJECT IDENTIFIER ::= {versionOne 9}
brainpoolP320t1 OBJECT IDENTIFIER ::= {versionOne 10}
brainpoolP384r1 OBJECT IDENTIFIER ::= {versionOne 11}
brainpoolP384t1 OBJECT IDENTIFIER ::= {versionOne 12}
brainpoolP512r1 OBJECT IDENTIFIER ::= {versionOne 13}
brainpoolP512t1 OBJECT IDENTIFIER ::= {versionOne 14}
```

4.5 Syntax for Elliptic Curve Private Keys

In [6] and [7] no ASN.1 syntax for elliptic curve private keys is given. Therefore, the ASN.1 syntax for elliptic curve private keys and the corresponding recommendations as presented in [13] are fully adopted:

```
ECGPrivateKey{CURVES:IOSet} ::= SEQUENCE {
 version     INTEGER {ecgPrivkeyVer1(1)}(ecgPrivkeyVer1),
 privateKey  OCTET STRING,
 parameters  [0] Parameters{{IOSet}} OPTIONAL,
 publicKey   [1] BIT STRING OPTIONAL
}
```

The components of `ECGPrivateKey{}` have the following meaning:

- The component `version` specifies the version number of the elliptic curve private key structure. In the syntax above, the `INTEGER` element `ecgPrivkeyVer1` is created, whose value is set to 1.

- The component `privateKey` is the private key defined to be the octet string of length $\lceil(\log_2 n)/8\rceil$, where n is the order of the curve. The octet string is obtained from the associated unsigned integer via the encoding presented in [6], Section 4.3.1.

- The optional component `parameters` specifies the elliptic curve domain parameters as defined in [6] associated to the private key. If the parameters are known from another context, then this component may be `NULL` or omitted.

- The optional component `publicKey` contains the elliptic curve public key associated with the given private key. This component may be `NULL` or omitted.

According to [13], the syntax for `ECGPrivateKey{}` may be used to convey elliptic curve private keys using the syntax for `PrivateKeyInfo` as defined in PKCS#8 [11]. For an instance of `PrivateKeyInfo`, the following recommendations of [13] are given:

- The value of the component `privateKeyAlgorithm` within `PrivateKeyInfo` shall be `ecgPublicKey` as defined in Section 4.4.

- The elliptic curve domain parameters shall be placed in the `privateKeyAlgorithm` field of `PrivateKeyInfo`, and the `parameters` field of `ECGPrivateKey{}` shall be omitted.

4.6 Syntax for Digital Signatures

This section provides the syntax for the digital signatures defined in this document.

The X.509 certificate and certificate revocation list (CRL) types include an ASN.1 algorithm object identifier to identify the signature type and format. When ECGDSA and one of the hash functions RIPEMD-160, SHA-1 (= SHA-160), SHA-224, SHA-256, SHA-384, and SHA-512 is used to sign an X.509 certificate or CRL, the signature shall be identified by the corresponding object identifier, as defined below:

```
ecgSignature OBJECT IDENTIFIER ::= {ecgDsaStd signatures(4)}

ecgSignatureWithripemd160 OBJECT IDENTIFIER ::= {ecgSignature 1}
ecgSignatureWithsha1      OBJECT IDENTIFIER ::= {ecgSignature 2}
ecgSignatureWithsha224    OBJECT IDENTIFIER ::= {ecgSignature 3}
ecgSignatureWithsha256    OBJECT IDENTIFIER ::= {ecgSignature 4}
ecgSignatureWithsha384    OBJECT IDENTIFIER ::= {ecgSignature 5}
ecgSignatureWithsha512    OBJECT IDENTIFIER ::= {ecgSignature 6}
```

For example, a signature generated with ECGDSA and hash function SHA-1 (= SHA-160) shall be identified by object identifier `ecgSignatureWithsha1`.

When a digital signature is identified by one of the object identifiers above, the signature shall be ASN.1 encoded using the syntax:

```
ECGDSA-Sig-Value ::= SEQUENCE {
 r  INTEGER,
 s  INTEGER
}
```

X.509 certificates and CRLs represent signatures as a value of type `BIT STRING`. In these cases, the entire encoding of a value of type `ECGDSA-Sig-Value` shall be the value of the bit string (including tag and length field in case DER-encoding is being used).

4.7 ASN.1 Module

The following ASN.1 module contains all of the syntax defined in this document. Most parts of the syntax are included from [6] by using the `IMPORTS` clause:

```
TTT-ECGDSA {iso(1) identified-organization(3) teletrust(36) algorithm(3)
            signature-algorithm(3) ecSign(2) ecgDsa(5) module(5) 1}

DEFINITIONS EXPLICIT TAGS ::= BEGIN

--
-- EXPORTS All;
--

IMPORTS FieldID{}, prime-field, characteristic-two-field, Prime-p,
        Trinomial, Pentanomial, gnBasis, tpBasis, ppBasis, ECPoint,
        CHARACTERISTIC-TWO, ECParameters, primeCurve, CURVES FROM ANSI-X9-62
        Parameters{}, ALGORITHM, AlgorithmIdentifier{},
        ANSICurveNames FROM ANSI-X9-63;

ecgDsaStd OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) teletrust(36)
  algorithm(3) signature-algorithm(3) ecSign(2) ecgDsa(5)}

FIELD-ID ::= TYPE-IDENTIFIER  -- ISO/IEC 8824-2:1995(E), Annex A

-- should by defined in ANSI-X9-63
FieldTypes FIELD-ID ::= {
{ Prime-p IDENTIFIED BY prime-field } |
{ Characteristic2Set IDENTIFIED BY characteristic-two-field },
 ...
}
-- should by defined in ANSI-X9-63
Characteristic2Set ::= Characteristic-two { {F2mBasisTypes} }

-- should by defined in ANSI-X9-63
Characteristic-two { CHARACTERISTIC-TWO:IOSet } ::= SEQUENCE {
 m INTEGER,
 basis CHARACTERISTIC-TWO.&id({F2mBasisTypes}),
 parameters CHARACTERISTIC-TWO.&Type({F2mBasisTypes}{@basis})
}

F2mBasisTypes CHARACTERISTIC-TWO ::= {
```

```
 { NULL IDENTIFIED BY gnBasis } |
 { Trinomial IDENTIFIED BY tpBasis } |
 { Pentanomial IDENTIFIED BY ppBasis }|
 { IrrPolynomial IDENTIFIED BY ipBasis },
 ...
}

-- polynomial basis representation (different from trinomial
-- and pentanomial basis) of F2^m
-- reduction polynomial f(x) represented as field element
-- (f(x) mod x^m) of F2^m

FieldElement ::= OCTET STRING
IrrPolynomial ::= FieldElement

ecgFieldType OBJECT IDENTIFIER ::= {ecgDsaStd fieldType(1)}

characteristicTwoField OBJECT IDENTIFIER ::= {ecgFieldType 1}

characteristicTwoBasis OBJECT IDENTIFIER ::=
 {characteristicTwoField basisType(1)}

ipBasis OBJECT IDENTIFIER ::= {characteristicTwoBasis 1}

ecgSignature OBJECT IDENTIFIER ::= {ecgDsaStd signatures(4)}

ecgSignatureWithripemd160 OBJECT IDENTIFIER ::= {ecgSignature 1}
ecgSignatureWithsha1      OBJECT IDENTIFIER ::= {ecgSignature 2}
ecgSignatureWithsha224    OBJECT IDENTIFIER ::= {ecgSignature 3}
ecgSignatureWithsha256    OBJECT IDENTIFIER ::= {ecgSignature 4}
ecgSignatureWithsha384    OBJECT IDENTIFIER ::= {ecgSignature 5}
ecgSignatureWithsha512    OBJECT IDENTIFIER ::= {ecgSignature 6}

ecgKeyType OBJECT IDENTIFIER ::= {ecgDsaStd keyType(2)}
ecgPublicKey OBJECT IDENTIFIER ::= {ecgKeyType publicKey(1)}

ecgPublicKeyType ALGORITHM ::= {
OID ecgPublicKey PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-RIPEMD160 ALGORITHM ::= {
OID ecgSignatureWithripemd160 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA1 ALGORITHM ::= {
OID ecgSignatureWithsha1 PARMS Parameters {{ExtendedCurveNames}}
}
```

```
ecgdsa-SHA224 ALGORITHM ::= {
OID ecgSignatureWithsha224 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA256 ALGORITHM ::= {
OID ecgSignatureWithsha256 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA384 ALGORITHM ::= {
OID ecgSignatureWithsha384 PARMS Parameters {{ExtendedCurveNames}}
}

ecgdsa-SHA512 ALGORITHM ::= {
OID ecgSignatureWithsha512 PARMS Parameters {{ExtendedCurveNames}}
}

ECGPKAlgorithms ALGORITHM ::= {
 ecgPublicKeyType | ecgdsa-RIPEMD160 |
 ecgdsa-SHA1      | ecgdsa-SHA224    |
 ecgdsa-SHA256    | ecgdsa-SHA384    |
 ecgdsa-SHA512,
 ...
}

SubjectPublicKeyInfo ::= SEQUENCE {
 algorithm AlgorithmIdentifier {{ECGPKAlgorithms}},
 subjectPublicKey BIT STRING
}

BrainpoolCurveNames CURVES ::= {
 {ID brainpoolP160r1} |
 {ID brainpoolP160t1} |
 {ID brainpoolP192r1} |
 {ID brainpoolP192t1} |
 {ID brainpoolP224r1} |
 {ID brainpoolP224t1} |
 {ID brainpoolP256r1} |
 {ID brainpoolP256t1} |
 {ID brainpoolP320r1} |
 {ID brainpoolP320t1} |
 {ID brainpoolP384r1} |
 {ID brainpoolP384t1} |
 {ID brainpoolP512r1} |
 {ID brainpoolP512t1},
 ...
}

ExtendedCurveNames CURVES ::= {
```

```
 { ANSICurveNames } |
 { BrainpoolCurveNames },
 ...
}

-- for future use
-- ecg-curves OBJECT IDENTIFIER ::= {ecgDsaStd curves(3)}

ecc-brainpool OBJECT IDENTIFIER ::= {
  iso(1) identified-organization(3) teletrust(36) algorithm(3)
  signature-algorithm(3) ecSign(2) ecStdCurvesAndGeneration(8)}

ellipticCurve OBJECT IDENTIFIER ::= {ecc-brainpool 1}

versionOne OBJECT IDENTIFIER ::= {ellipticCurve 1}

brainpoolP160r1 OBJECT IDENTIFIER ::= {versionOne 1}
brainpoolP160t1 OBJECT IDENTIFIER ::= {versionOne 2}
brainpoolP192r1 OBJECT IDENTIFIER ::= {versionOne 3}
brainpoolP192t1 OBJECT IDENTIFIER ::= {versionOne 4}
brainpoolP224r1 OBJECT IDENTIFIER ::= {versionOne 5}
brainpoolP224t1 OBJECT IDENTIFIER ::= {versionOne 6}
brainpoolP256r1 OBJECT IDENTIFIER ::= {versionOne 7}
brainpoolP256t1 OBJECT IDENTIFIER ::= {versionOne 8}
brainpoolP320r1 OBJECT IDENTIFIER ::= {versionOne 9}
brainpoolP320t1 OBJECT IDENTIFIER ::= {versionOne 10}
brainpoolP384r1 OBJECT IDENTIFIER ::= {versionOne 11}
brainpoolP384t1 OBJECT IDENTIFIER ::= {versionOne 12}
brainpoolP512r1 OBJECT IDENTIFIER ::= {versionOne 13}
brainpoolP512t1 OBJECT IDENTIFIER ::= {versionOne 14}

ECGDSA-Sig-Value ::= SEQUENCE {
 r INTEGER,
 s INTEGER
}

-- private-key syntax adopted from:
--     SEC 1: Elliptic Curve Cryptography, Certicom Research, Version 1.0

ECGPrivateKey{CURVES:IOSet} ::= SEQUENCE {
 version INTEGER {ecPrivkeyVer1(1)}(ecPrivkeyVer1),
 privateKey OCTET STRING,
 parameters [0] Parameters{{IOSet}} OPTIONAL,
 publicKey [1] BIT STRING OPTIONAL
}

END
```

Bibliography

[1] FIPS 180-2, *Secure hash standard*, Federal Information Processing Standards Publication 180-2, August 2002, Including the change notice of February 2004 specifying SHA-22. Available at `http://csrc.nist.gov/`.

[2] G.B. Agnew, R.C. Mullin, and S.A. Vanstone, *Improved digital signature schemes based on discrete exponentiation*, Electronic Letters (1990), no. 26, 1024–1025.

[3] ECC Brainpool, *ECC Brainpool Standard Curves and Curve Generation version 1.0*, Tech. report, TeleTrusT Deutschland e.V., 2005, Available at `http://www.ecc-brainpool.org/download/BP-Kurven-aktuell.pdf`.

[4] H. Dobbertin, A. Bosselaers, and B. Preneel, *RIPEMD-160, a strengthened version of RIPEMD*, Fast Software Encryption (D. Gollmann, ed.), LNCS, no. 1039, Springer-Verlag, 1996, Available at `http://www.esat.kuleuven.ac.be/~cosicart/ps/AB-9601/AB-9601.ps.gz`, pp. 71–82.

[5] T. ElGamal, *A public-key cryptosystem and a signature scheme based on discrete logarithms*, IEEE Trans. on Information Theory **IT 31** (1985), no. 4, 469–472.

[6] American National Standard for Financial Services, *ANSI X9.62-1998: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American National Standards Institute, 1998.

[7] ———, *ANSI X9.63-2001: Public Key Cryptography for the Financial Services Industry: Key Agreement and Key Transport Using Elliptic Curve Cryptography*, American National Standards Institute, 2001.

[8] ———, *ANSI X9.62-2005: Public Key Cryptography for the Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)*, American National Standards Institute, 2005, replaces *ANSI X9-62*, (1998).

[9] International Telecomunication Union, *ITU-T X.681: Information technology – Abstract Syntax Notation One (ASN.1): Information object specification*, 2002.

[10] N. Koblitz, *Elliptic curve cryptosystems*, Math. Comp. (1987), no. 48, 203–209.

[11] RSA Laboratories, *PKCS#8 v1.2: Private-Key Information Syntax Standard*, RSA Laboratories, 1993.

[12] V. Miller, *Use of elliptic curves in cryptology*, Proceedings of CRYPTO '85, Lecture Notes in Computer Science, vol. 218, Springer-Verlag, 1986, pp. 417–425.

[13] Certicom Research, *Standards For Efficient Cryptography – SEC 1: Elliptic Curve Cryptography, Version 1.0*, 2000, Available at `http://www.secg.org/`.