GUIDELINES FOR EFFICIENT CRYPTOGRAPHY

GEC 2: Test Vectors for SEC 1

Certicom Research

Contact: Simon Blake-Wilson (sblakewilson@certicom.com)

Hugh MacDonald (hmacdonald@certicom.com)

Minghua Qu (mqu@certicom.com)

Working Draft
September, 1999
Version 0.3

Contents

1 Introduction

1.1 Overview

This document presents test vectors for the cryptographic schemes specified in SEC 1 [1]. For each scheme in SEC 1, an example using elliptic curve domain parameters over $\mathbb{F}_p$ is described, and an example using elliptic curve domain parameters over $\mathbb{F}_{2^m}$ is described. For ECAES and ECDH, examples using both the standard and the cofactor Diffie-Hellman primitive are described.

1.2 Aim

The test vectors presented in this document are meant to assist implementors of SEC 1 in checking that they have implemented the standard correctly.

The test vectors show intermediate steps as well as the result of the operation of each scheme to further enable implementors to locate errors in the event of their implementation not agreeing with the test vectors.

1.3 Organization

This document is organized as follows.

Each section of the document gives test vectors for a different protocol. Section 2 gives the test vectors for ECDSA. Section 3 gives the test vectors for ECAES. Section 4 gives the test vectors for ECDH. Section 5 gives the test vectors for ECMQV. Finally Section 6 lists the references cited in the document.

Within each section, the first subsection shows the test vectors over $\mathbb{F}_p$, while the second subsection shows the test vectors over $\mathbb{F}_{2^m}$. In the case of ECAES and ECDH there is a third subsection which shows test vectors over $\mathbb{F}_{2^m}$ using the cofactor Diffie-Hellman primitive.

2 Test Vectors for ECDSA

This section provides test vectors for ECDSA as specified in Section 4.1 of SEC 1 [1]. Section 2.1 provides test vectors for ECDSA using elliptic curve domain parameters over $\mathbb{F}_p$, and Section 2.2 provides test vectors for ECDSA using elliptic curve domain parameters over $\mathbb{F}_{2^m}$.

2.1 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_p$

This section provides test vectors for ECDSA using elliptic curve domain parameters over $\mathbb{F}_p$. U and V use ECDSA as follows.

2.1.1 Scheme Setup

U decides to use ECDSA with the hash function SHA-1 and the elliptic curve domain parameters secp160r1 specified in GEC 1 [2]. U conveys this information to V.

2.1.2 Key Deployment for U

U selects a key pair (d_U, Q_U) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: U selects a key pair.

1. Generate an integer d_U.

   1.1. Randomly or pseudorandomly select an integer d_U in the interval $[1, n-1]$.

$$d_U = 971761939728640320549601132085879836204587084162$$

   1.2. Convert d_U to the octet string $\overline{d_U}$.

$$\overline{d_U} = \texttt{AA374FFC 3CE144E6 B0733079 72CB6D57 B2A4E982}$$

2. Calculate $Q_U = (x_U, y_U) = d_U \times G$.

$$x_U = 466448783855397898016055842232266600516272889280$$
$$y_U = 1110706324081757720403272427311003102474457754220$$

As an octet string with point compression, we have:

$$\overline{Q_U} = \texttt{02 51B4496F ECC406ED 0E75A24A 3C032062 51419DC0}$$

Output: The elliptic curve key pair (d_U, Q_U) with:

$$d_U = 971761939728640320549601132085879836204587084162$$
$$Q_U = (466448783855397898016055842232266600516272889280, 1110706324081757720403272427311003102474457754220)$$

U shares Q_U with V in an authentic manner. V should check that Q_U is valid.

2.1.3 Signing Operation for U

Suppose U wants to convey the message M = "abc" to V. U signs M as follows.

Input: The octet string $M = \texttt{616263}$ which represents the message "abc".

Actions: U signs M.

1. Select an ephemeral key pair (k, R) as follows using the key pair generation primitive specified in Section 3.2.1 of SEC 1 [1].

   1.1. Randomly or pseudorandomly select an integer k in the interval $[1, n-1]$.

   $$k = 702232148019446860144825009548118511996283736794$$

   1.2. Compute $R = (x_R, y_R) = k \times G$.

   $$x_R = 1176954224688105769566774212902092897866168635793$$
   $$y_R = 1130322298812061698910820170565981471918861336822$$

2. Convert the field element x_R to an integer using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$\overline{x_R} = 1176954224688105769566774212902092897866168635793$$

3. Derive an integer r from $\overline{x_R}$.

   3.1. Set $r \equiv \overline{x_R} \pmod{n}$.

$$r = 1176954224688105769566774212902092897866168635793$$

   3.2. $r \neq 0$, OK.

   3.3. r is represented as the octet string $\overline{r}$.

$$\overline{r} = \texttt{CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191}$$

4. SHA-1 is applied to M to get $H = SHA\text{-}1(M)$.

$$H = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

5. Derive an integer e from H.

   5.1. Convert the octet string H to a bit string $\overline{H}$ using the conversion routine specified in Section 2.3.2 of SEC 1 [1].

$$\overline{H} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110 10000001 01101010 10111010 00111110 00100101 01110001 01111000 01010000 11000010 01101100 10011100 11010000 11011000 10011101}$$

   5.2. Set $\overline{E} = \overline{H}$ since $\log_2 n \geq 8hashlen$.

$$\overline{E} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110 10000001 01101010 10111010 00111110 00100101 01110001 01111000 01010000 11000010 01101100 10011100 11010000 11011000 10011101}$$

   5.3. Convert $\overline{E}$ to an octet string E using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$E = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

5.4. Convert E to an integer e using the conversion routine specified in Section 2.3.8 of SEC 1 [1].

$$e = 968236873715988614170569073515315707566766479517$$

6. Compute the integer s.

6.1. Compute $s \equiv k^{-1}(e + d_U \cdot r) \pmod{n}$.

$$s = 299742580584132926933316745664091704165278518100$$

6.2. $s \neq 0$, OK.

6.3. s is represented as the octet string, $\overline{s}$, where:

$$\overline{s} = \texttt{3480EC13 71A091A4 64B31CE4 7DF0CB8A A2D98B54}$$

Output: The signature $S = (r, s)$.

$$r = 1176954224688105769566774212902092897866168635793$$
$$s = 299742580584132926933316745664091704165278518100$$

or as octet strings:

$$\overline{r} = \texttt{CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191}$$
$$\overline{s} = \texttt{3480EC13 71A091A4 64B31CE4 7DF0CB8A A2D98B54}$$

U conveys the signed message consisting of M and (r, s) to V.

2.1.4 Verifying Operation for V

V verifies the signed message from U as follows.

Input: The verifying operations takes the following input.

1. The octet string $M = \texttt{616263}$ which represents the message “abc”.

2. U's purported signature $S = (r, s)$ on M.

$$r = 1176954224688105769566774212902092897866168635793$$
$$s = 299742580584132926933316745664091704165278518100$$

Actions: V verifies the signature on the message M.

1. r and s are both integers in the interval $[1, n-1]$, OK.
2. SHA-1 is applied to M to get $H = SHA\text{-}1(M)$.

$$H = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

3. Derive an integer e from H.

3.1. Convert the octet string H to a bit string $\overline{H}$ using the conversion routine specified in Section 2.3.2 of SEC 1 [1].

$$\overline{H} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110 10000001 01101010 10111010 00111110 00100101 01110001 01111000 01010000 11000010 01101100 10011100 11010000 11011000 10011101}$$

3.2. Set $\overline{E} = \overline{H}$ since $\log_2 n \geq 8hashlen$.

$$\overline{E} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110 10000001 01101010 10111010 00111110 00100101 01110001 01111000 01010000 11000010 01101100 10011100 11010000 11011000 10011101}$$

3.3. Convert $\overline{E}$ to an octet string E using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$E = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

3.4. Convert E to an integer e using the conversion routine specified in Section 2.3.8 of SEC 1 [1].

$$e = 968236873715988614170569073515315707566766479517$$

4. Compute $u_1 \equiv es^{-1} \pmod n$ and $u_2 \equiv rs^{-1} \pmod n$.

$$u_1 = 1264923452375560418053904424459712465512263948 66$$
$$u_2 = 642136937233451268764953375477669732399252982122$$

5. Compute $R = (x_R, y_R) = u_1G + u_2Q_U$.

5.1. Compute $u_1G = (x_{U_1}, y_{U_1})$.

$$x_{U_1} = 559637225459801172484164154368876326912482639549$$
$$y_{U_1} = 1427364757892877133166464896740210315153233662312$$

5.2. Compute $u_2Q_U = (x_{U_2}, y_{U_2})$.

$$x_{U_2} = 1096326382299378890940501642113021093797486469420$$
$$y_{U_2} = 1361206527591198621565826173236094337930170472426$$

5.3. Compute $R = (x_R, y_R) = u_1G + u_2Q_U$.

$$x_R = 1176954224688105769566774212902092897866168635793$$
$$y_R = 1130322298812061698910820170565981471918861336822$$

5.4. $R \neq \mathcal{O}$, OK.

6. Convert the field element x_R to an integer using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$\overline{x_R} = 1176954224688105769566774212902092897866168635793$$

7. Set $v \equiv \overline{x_R} \pmod n$.

$$v = 1176954224688105769566774212902092897866168635793$$

8. $v = r$, OK.

Output: 'Valid' to indicate that the signed message is valid.

2.2 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$

This section provides test vectors for ECDSA using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$. U and V use ECDSA as follows.

2.2.1 Scheme Setup

U decides to use ECDSA with the hash function SHA-1 and the elliptic curve domain parameters sect163k1 specified in GEC 1 [2]. U conveys this information to V.

2.2.2 Key Deployment for U

U selects a key pair (d_U, Q_U) as follows using the key generation primitive specified Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: U selects a key pair.

1. Generate the integer d_U.

   1.1. Randomly or pseudorandomly select an integer d_U in the interval $[1, n-1]$.

   $$d_U = 5321230001203043918714616464614664646674949479949$$

   1.2. Convert d_U to the octet string $\overline{d_U}$.

   $$\overline{d_U} = \texttt{03 A41434AA 99C2EF40 C8495B2E D9739CB2 155A1E0D}$$

2. Calculate $Q_U = (x_U, y_U) = d_U \times G$.

   $$x_U = \texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$
   $$y_U = \texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776}$$

   As an octet string with point compression we have:

$$\overline{Q_U} = \texttt{0303 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$

Output: The elliptic curve key pair (d_U, Q_U) with:

$$d_U = 532123000120304391871461646461466464667494947 9949$$
$$Q_U = (\texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC},$$
$$\texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776})$$

U shares Q_U with V in an authentic manner. V should check that Q_U is valid.

2.2.3 Signing Operation for U

Suppose U wants to convey the message M = "abc" to V. U signs M as follows.

Input: The octet string $M = \texttt{616263}$ which represents the message "abc".

Actions: U signs M.

1. Select an ephemeral key pair (k, R) as follows using the key pair generation primitive specified in Section 3.2.1 of SEC 1 [1].

   1.1. Randomly or pseudorandomly select an integer k in the interval $[1, n-1]$.

   $$k = 936523985789236956265265265235675811949404040044$$

   1.2. Compute $R = (x_R, y_R) = k \times G$.

   $$x_R = \texttt{04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B}$$
   $$y_R = \texttt{03 1FC936D7 3163B858 BBC5326D 77C19839 46405264}$$

2. Convert the field element x_R to an integer using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

   $$\overline{x_R} = 6721203149925103462794551781766000547003321473691$$

3. Derive an integer r from $\overline{x_R}$.

3.1. Set $r \equiv \overline{x_R} \pmod{n}$.

$$r = 875196600601491789979810028167552198674202899628$$

3.2. $r \neq 0$, OK.

3.3. r is represented as the octet string $\overline{r}$.

$$\overline{r} = \texttt{994D2C41 AA30E529 52AEA846 2370471B 2B0A34AC}$$

4. SHA-1 is applied to M to get $H = SHA\text{-}1(M)$.

$$H = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

5. Derive an integer e from H.

5.1. Convert the octet string H to a bit string $\overline{H}$ using the conversion routine specified in Section 2.3.2 of SEC 1 [1].

$$\overline{H} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110}$$
$$\texttt{10000001 01101010 10111010 00111110 00100101 01110001}$$
$$\texttt{01111000 01010000 11000010 01101100 10011100 11010000}$$
$$\texttt{11011000 10011101}$$

5.2. Set $\overline{E} = \overline{H}$ since $\log_2 n \geq 8hashlen$.

$$\overline{E} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110}$$
$$\texttt{10000001 01101010 10111010 00111110 00100101 01110001}$$
$$\texttt{01111000 01010000 11000010 01101100 10011100 11010000}$$
$$\texttt{11011000 10011101}$$

5.3. Convert $\overline{E}$ to an octet string E using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$E = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

5.4. Convert E to an integer e using the conversion routine specified in Section 2.3.8 of SEC 1 [1].

$$e = 968236873715988614170569073515315707566766479517$$

6. Compute the integer s.

6.1. Compute $s \equiv k^{-1}(e + d_U r) \pmod{n}$.

$$s = 1935199835333115956886966454901154618180070051199$$

6.2. $s \neq 0$, OK.

6.3. s is represented as the octet string, $\overline{s}$, where:

$$\overline{s} = \texttt{01 52F95CA1 5DA1997A 8C449E00 CD2AA2AC CB988D7F}$$

Output: The signature $S = (r, s)$.

$$r = 875196600601491789979810028167552198674202899628$$
$$s = 1935199835333115956886966454901154618180070051199$$

or as octet strings:

$$\overline{r} = \texttt{994D2C41 AA30E529 52AEA846 2370471B 2B0A34AC}$$
$$\overline{s} = \texttt{01 52F95CA1 5DA1997A 8C449E00 CD2AA2AC CB988D7F}$$

U conveys the signed message consisting of M and (r, s) to V.

2.2.4 Verifying Operation for V

V verifies the message from U as follows.

Input: The verifying operation takes the following input.

1. The octet string $M = \texttt{616263}$ which represents the message “abc”.

2. U's purported signature $S = (r, s)$ on M.

$$r = 875196600601491789979810028167552198674202899628$$
$$s = 1935199835333115956886966454901154618180070051199$$

Actions: V verifies the signature on the message M.

1. r and s are both integers in the interval $[1, n-1]$, OK.
2. SHA-1 is applied to M to get $H = SHA\text{-}1(M)$.

$$H = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

3. Derive an integer e from H.

3.1. Convert the octet string H to a bit string $\overline{H}$ using the conversion routine specified in Section 2.3.2 of SEC 1 [1].

$$\overline{H} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110}$$
$$\texttt{10000001 01101010 10111010 00111110 00100101 01110001}$$
$$\texttt{01111000 01010000 11000010 01101100 10011100 11010000}$$
$$\texttt{11011000 10011101}$$

3.2. Set $\overline{E} = \overline{H}$ since $\log_2 n \geq 8hashlen$.

$$\overline{E} = \texttt{10101001 10011001 00111110 00110110 01000111 00000110}$$
$$\texttt{10000001 01101010 10111010 00111110 00100101 01110001}$$
$$\texttt{01111000 01010000 11000010 01101100 10011100 11010000}$$
$$\texttt{11011000 10011101}$$

3.3. Convert $\overline{E}$ to an octet string E using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$E = \texttt{A9993E36 4706816A BA3E2571 7850C26C 9CD0D89D}$$

3.4. Convert E to an integer e using the conversion routine specified in Section 2.3.8 of SEC 1 [1].

$$e = 968236873715988614170569073515315707566766479517$$

4. Compute $u_1 \equiv es^{-1} \pmod{n}$ and $u_2 \equiv rs^{-1} \pmod{n}$.

$$u_1 = 565806754829218233303449435097509340497193031129 8$$

$$u_2 = 2390570840421010673757367220187439778211658217319$$

5. Compute $R = (x_R, y_R) = u_1G + u_2Q_U$.

5.1. Compute $u_1G = (x_{U_1}, y_{U_1})$.

x_{U_1} = `05 1B4B9235 90399545 34D77469 AC7434D7 45BE784D`

y_{U_1} = `01 C657D070 935987CA 79976B31 6ED2F533 41058956`

5.2. Compute $u_2Q_U = (x_{U_2}, y_{U_2})$.

x_{U_2} = `07FD04AF 05DCAF73 39F6F89C 52EF27FE 94699AED`

y_{U_2} = `AA84BE48 C0F1256F A31AAADD F4ADDDD5 AD1F0E14`

5.3. Compute $R = (x_R, y_R) = u_1G + u_2Q_U$.

x_R = `04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B`

y_R = `03 1FC936D7 3163B858 BBC5326D 77C19839 46405264`

5.4. $R \neq O$, OK.

6. Convert the field element x_R to an integer using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$\overline{x_R} = 6721203149925103462794551781766000547003321473691$$

7. Set $v \equiv \overline{x_R} \pmod{n}$.

$$v = 875196600601491789979810028167552198674202899628$$

8. $v = r$, OK.

Output: ‘Valid’ to indicate that the signed message is valid.

3 Test Vectors for ECAES

This section provides test vectors for ECAES as specified in Section 5.1 of SEC 1 [1]. Section 3.1 provides test vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_p$, Section 3.2 provides test vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_{2^m}$ and the standard Diffie-Hellman primitive, and Section 3.3 provides test vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_{2^m}$ and the cofactor Diffie-Hellman primitive.

3.1 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_p$

This section provides text vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_p$. U and V use ECAES as follows.

3.1.1 Scheme Setup

V decides to use ECAES with the key derivation function ANSI-X9.63-KDF with SHA-1 specified in Section 3.6 of SEC 1 [1], the MAC scheme HMAC-SHA-1-160 with 20 octet keys, the XOR symmetric encryption scheme and the elliptic curve domain parameters secp160r1 specified in GEC 1 [2]. V conveys these decisions to U. U decides to represent elliptic curve points in compressed form.

Note that in this case, the cofactor is $h = 1$, so the choice between the standard and cofactor Diffie-Hellman primitives is unnecessary.

3.1.2 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.

   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

$$d_V = 399525573676508631577122671218044116107572676710$$

   1.2. Convert d_V to the octet string $\overline{d_V}$.

$$\overline{d_V} = \texttt{45FB58A9 2A17AD4B 15101C66 E74F277E 2B460866}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

$$x_V = 420773078745784176406965940076771545932416607676$$
$$y_V = 221937774842090227911893783570676792435918278531$$

As an octet string with point compression we have:

$$\overline{Q_V} = \texttt{03 49B41E0E 9C0369C2 328739D9 0F63D567 07C6E5BC}$$

Output: The elliptic curve key pair (d_V, Q_V) with:

$$d_V = 399525573676508631577122671218044116107572676710$$
$$Q_V = (420773078745784176406965940076771545932416607676,$$
$$221937774842090227911893783570676792435918278531)$$

V shares Q_V with U in an authentic manner. U should check that Q_V is valid.

3.1.3 Encryption Operation for U

Suppose U wants to convey the message M “abcdefghijklmnopqrst” confidentially to V.

Input: The encryption operation takes the following input.

1. The octet string $M =$ `61626364 65666768 696A6B6C 6D6E6F70 71727374` which represents the message “abcdefghijklmnopqrst”.
2. The optional strings $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: U encrypts the message M.

1. Select an ephemeral key pair (k, R) as follows using the key pair generation primitive specified in Section 3.2.1 of SEC 1 [1].

   1.1. Randomly or pseudorandomly select an integer k in the interval $[1, n-1]$.

$$k = 702232148019446860144825009548118511996283736794$$

1.2. Compute $R = (x_R, y_R) = k \times G$.

$$x_R = 1176954224688105769566774212902092897866168635793$$
$$y_R = 1130322298812061698910820170565981471918861336822$$

2. Convert the point R to an octet string $\overline{R}$ with point compression using the conversion routine specified in Section 2.3.3 of SEC 1 [1].

2.1. Convert x_R to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

$\overline{x_R}$ = `CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191`

2.2. Get $\overline{R}$.

$\overline{R}$ = `02 CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191`

3. Compute the shared secret field element z using the standard elliptic curve Diffie Hellman primitive specified in Section 3.3.1 of SEC 1 [1].

3.1. Compute $P = (x_P, y_P) = k \times Q_V$.

$$x_P = 171537086520105273255189335256955712560931509051$$
$$y_P = 848085177066589686397671271789061798084202394410$$

3.2. $P \neq O$, OK.

3.3. Set $z = x_P$.

$$z = 171537086520105273255189335256955712560931509051$$

4. Convert z to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

Z = `1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B`

5. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

5.1. Append $Counter_1 = $ `00000001` to the right of Z.

$$Z_1 = \texttt{1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B 00000001}$$

5.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699}$$

5.3. Append $Counter_2 = $ `00000002` to the right of Z.

$$Z_2 = \texttt{1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B 00000002}$$

5.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD}$$

5.5. Get $K = Hash_1 || Hash_2$.

$$K = \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699}$$
$$\texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD}$$

6. Get EK and MK.

6.1. Get EK from K.

$$EK = \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699}$$

6.2. Get MK from K.

$$MK = \texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD}$$

7. Encrypt the octet string M under EK to produce the ciphertext EM using the XOR encryption scheme as specified in Section 3.8.3 of SEC 1 [1].

7.1. Convert M to a bit string $M_0M_1M_2 \ldots M_{159}$.

$$\overline{M} = \texttt{01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000 01101001 01101010 01101011 01101100 01101101 01101110 01101111 01110000 01110001 01110010 01110011 01110100}$$

7.2. Convert EK to a bit string $EK_0EK_1EK_2 \ldots EK_{159}$.

$$\overline{EK} = \texttt{00010000 01000001 10101011 00010100 11000110 01111100 11100110 10000010 00011100 11101001 01000010 01100001 01110110 11001111 00010100 10111000 00000100 11100110 01000110 10011001}$$

7.3. Use XOR to encrypt the M by $\overline{M} \oplus \overline{EK}$.

$$\overline{EM} = \texttt{01110001 00100011 11001000 01110000 10100011 00011010 10000001 11101010 01110101 10000011 00101001 00001101 00011011 10100001 01111011 11001000 01110101 10010100 00110101 11101101}$$

7.4. Convert $\overline{EM}$ to an octet string EM using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$EM = \texttt{7123C870 A31A81EA 7583290D 1BA17BC8 759435ED}$$

8. Compute the tag D on EM under MK using the MAC sceme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.3 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\overline{EM} = \texttt{01110001 00100011 11001000 01110000 10100011 00011010 10000001 11101010 01110101 10000011 00101001 00001101 00011011 10100001 01111011 11001000 01110101 10010100 00110101 11101101}$$

8.2. Convert MK to a bit string.

$$\overline{MK} = \texttt{10010011 11001011 11001110 10111100 10100100 00011001 11111101 01011101 01011000 00101110 00000011 10010100 01111110 00100001 11011000 01111001 01100111 01110000 10101111 11111101}$$

8.3. Calculate the tag $D = MAC_{\overline{MK}}(\overline{EM})$ using the MAC sceme HMAC-SHA-1-160 with 20 octet keys.

$$D = \texttt{1CCDA9EB 4ED27360 BE896729 AD185493 622591E5}$$

Output: The ciphertext $C = \overline{R}||EM||D$.

$$C = \texttt{02 CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191 7123C870 A31A81EA 7583290D 1BA17BC8 759435ED 1CCDA9EB 4ED27360 BE896729 AD185493 622591E5}$$

U conveys the encrypted message C to V.

3.1.4 Decryption Operation for V

V decrypts the ciphertext C as follows.

Input: The decryption operation takes the following input.

1. The octet string C which is the ciphertext.

$$C = \texttt{02 CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191 7123C870 A31A81EA 7583290D 1BA17BC8 759435ED 1CCDA9EB 4ED27360 BE896729 AD185493 622591E5}$$

2. The optional inputs $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: V decrypts the message.

1. Parse C to get $\overline{R}, EM$, and D.

$$\begin{aligned} \overline{R} &= \texttt{02 CE2873E5 BE449563 391FEB47 DDCBA2DC 16379191} \\ EM &= \texttt{7123C870 A31A81EA 7583290D 1BA17BC8 759435ED} \\ D &= \texttt{13B59208 75D7AA0D BF569543 A80CCE46 A4A4074D} \end{aligned}$$

2. Convert $\overline{R}$ to an elliptic curve point $R = (x_R, y_R)$.

$$\begin{aligned} x_R &= 1176954224688105769566774212902092897866168635793 \\ y_R &= 1130322298812061698910820170565981471918861336822 \end{aligned}$$

3. Validate R using the primitive specified in Section 3.2.2.1 of SEC 1 [1].

   3.1. Verify that $R \neq \mathcal{O}$, OK.

   3.2. Verify that R is a point on the curve, OK.

   3.3. Verify that $nR = \mathcal{O}$, OK.

4. Derive the shared secret field element z using the standard elliptic curve Diffie Hellman primitive specified in Section 3.3.1 of SEC 1 [1].

   4.1. Compute $P = (x_P, y_P) = d_V \times R$.

$$\begin{aligned} x_P &= 171537086520105273255189335256955712560931509051 \\ y_P &= 848085177066589686397671271789061798084202394410 \end{aligned}$$

   4.2. $P \neq \mathcal{O}$, OK.

   4.3. Set $z = x_P$.

$$z = 171537086520105273255189335256955712560931509051$$

5. Convert z to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

$$Z = \texttt{1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B}$$

6. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

6.1. Append $Counter_1 = $ 00000001 to the right of Z.

$$Z_1 = \texttt{1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B 00000001}$$

6.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699}$$

6.3. Append $Counter_2 = $ 00000002 to the right of Z.

$$Z_2 = \texttt{1E0BFD2E 66F97B3B E0343A6C D517DA9B A213933B 00000002}$$

6.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD}$$

6.5. Get $K = Hash_1||Hash_2$.

$$\begin{aligned} K = \ & \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699} \\ & \texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD} \end{aligned}$$

7. Get EK and MK.

7.1. Get EK from K.

$$EK = \texttt{1041AB14 C67CE682 1CE94261 76CF14B8 04E64699}$$

7.2. Get MK from K.

$$MK = \texttt{93CBCEBC A419FD5D 582E0394 7E21D879 6770AFFD}$$

8. Check the tag D on EM under MK using the MAC sceme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.4 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\overline{EM} = $$ `01110001 00100011 11001000 01110000 10100011 00011010`
`10000001 11101010 01110101 10000011 00101001 00001101`
`00011011 10100001 01111011 11001000 01110101 10010100`
`00110101 11101101`

8.2. Convert MK to a bit string.

$$\overline{MK} = $$ `10010011 11001011 11001110 10111100 10100100 00011001`
`11111101 01011101 01011000 00101110 00000011 10010100`
`01111110 00100001 11011000 01111001 01100111 01110000`
`10101111 11111101`

8.3. Calculate the tag $D' = MAC_{\overline{MK}}(\overline{EM})$ using the MAC sceme HMAC-SHA-1-160 with 20 octet keys.

$$D' = $$ `1CCDA9EB 4ED27360 BE896729 AD185493 622591E5`

8.4. $D' = D$, OK.

9. Decrypt the octet string EM under EK to produce M using the XOR encryption scheme as specified in Section 3.8.4 of SEC 1 [1].

9.1. Convert EM to a bit string $EM_0EM_1EM_2 \ldots EM_{159}$.

$$\overline{EM} = $$ `01110001 00100011 11001000 01110000 10100011 00011010`
`10000001 11101010 01110101 10000011 00101001 00001101`
`00011011 10100001 01111011 11001000 01110101 10010100`
`00110101 11101101`

9.2. Convert EK to a bit string $EK_0EK_1EK_2 \ldots EK_{159}$.

$$\overline{EK} = $$ `00010000 01000001 10101011 00010100 11000110 01111100`
`11100110 10000010 00011100 11101001 01000010 01100001`
`01110110 11001111 00010100 10111000 00000100 11100110`
`01000110 10011001`

9.3. Use XOR to decrypt EM by $\overline{EM} \oplus \overline{EK}$.

$$\overline{M} = \texttt{01100001 01100010 01100011 01100100 01100101 01100110}$$
$$\texttt{01100111 01101000 01101001 01101010 01101011 01101100}$$
$$\texttt{01101101 01101110 01101111 01110000 01110001 01110010}$$
$$\texttt{01110011 01110100}$$

9.4. Convert $\overline{M}$ to an octet string M using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

Output: The message M.

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

M represents the text string “abcdefghijklmnopqrst”.

3.2 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$ and the Standard Diffie-Hellman Primitive

This section provides test vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$ and the standard Diffie-Hellman primitive. U and V use ECAES as follows.

3.2.1 Scheme Setup

V decides to use ECAES with the key derivation function ANSI-X9.63-KDF with SHA-1 specified in Section 3.6 of SEC 1 [1], the MAC scheme HMAC-SHA-1-160 with 20 octet keys, the XOR symmetric encryption scheme, the standard Diffie-Hellman primitive, and the elliptic curve domain parameters sect163k1 specified in GEC 1 [2]. V conveys these decisions to U. U decides to represent elliptic curve points in compressed form.

3.2.2 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.

   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

   $$d_V = 501870566195266176721440888203272826969530834326$$

   1.2. Convert d_V to an octet string $\overline{d_V}$.

   $$\overline{d_V} = \texttt{57E8A78E 842BF4AC D5C315AA 0569DB17 03541D96}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

   $$x_V = \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$
   $$y_V = \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7}$$

   As an octet string with point compression we have:

   $$\overline{Q_V} = \texttt{0307 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$

Output: The elliptic curve key pair (d_V, Q_V) with:

$$d_V = 501870566195266176721440888203272826969530834326$$
$$Q_V = (\texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}, \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7})$$

V shares Q_V with U in an authentic manner. U should check that Q_V is valid.

3.2.3 Encryption Operation for U

Suppose U wants to convey the message $M =$ “abcdefghijklmnopqrst” confidentially to V.

Input: The encryption operation takes the following input.

1. The octet string $M =$ `61626364 65666768 696A6B6C 6D6E6F70 71727374` which represents the message “abcdefghijklmnopqrst”.
2. The optional strings $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: U encrypts the message M.

1. Select an ephemeral key pair (k, R) using the key pair generation primitive specified in Section 3.2.1 of SEC 1 [1].

   1.1. Randomly or pseudorandomly select an integer k in the interval $[1, n-1]$.

$$k = 936523985789236956265265265235675811949404040044$$

   1.2. Compute $R = (x_R, y_R) = k \times G$.

x_R = `04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B`

y_R = `03 1FC936D7 3163B858 BBC5326D 77C19839 46405264`

2. Convert the point R to an octet string $\overline{R}$ with point compression using the conversion routine specified in Section 2.3.3 of SEC 1 [1].

$\overline{R}$ = `0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B`

3. Compute the shared secret field element z using the standard elliptic curve Diffie Hellman primitive specified in Section 3.3.1 of SEC 1 [1].

   3.1. Compute $P = (x_P, y_P) = k \times Q_V$.

x_P = `04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881`

y_P = `07 7A8B0052 E8C622CC 3DCC0613 50500262 173EB44E`

   3.2. $P \neq O$, OK.

   3.3. Set $z = x_P$.

z = `04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881`

4. Convert z to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

$$Z = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881}$$

5. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

   5.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

$$Z_1 = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881 00000001}$$

   5.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$

   5.3. Append $Counter_2 = \texttt{00000002}$ to the right of Z.

$$Z_2 = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881 00000002}$$

   5.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

   5.5. Get $K = Hash_1 || Hash_2$.

$$K = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$
$$\texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

6. Get EK and MK.

   6.1. Get EK from K.

$$EK = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$

6.2. Get MK from K.

$$MK = \texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

7. Encrypt the octet string M under EK to produce the ciphertext EM using the XOR encryption scheme as specified in Section 3.8.3 of SEC 1 [1].

7.1. Convert M to a bit string $M_0M_1M_2\ldots M_{159}$.

$$\overline{M} = \texttt{01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000 01101001 01101010 01101011 01101100 01101101 01101110 01101111 01110000 01110001 01110010 01110011 01110100}$$

7.2. Convert EK to a bit string $EK_0EK_1EK_2\ldots EM_{159}$.

$$\overline{EK} = \texttt{00000011 11000110 00100010 10000000 11001000 10010100 11100001 00000011 11000110 10000000 10110001 00111100 11010100 10110100 10101110 01110100 00001010 01011110 11110000 11000111}$$

7.3. Use XOR to encrypt M by $\overline{M} \oplus \overline{EK}$.

$$\overline{EM} = \texttt{01100010 10100100 01000001 11100100 10101101 11110010 10000110 01101011 10101111 11101010 11011010 01010000 10111001 11011010 11000001 00000100 01111011 00101100 10000011 10110011}$$

7.4. Convert $\overline{EM}$ to an octet string EM using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$EM = \texttt{62A441E4 ADF2866B AFEADA50 B9DAC104 7B2C83B3}$$

8. Compute the tag D on EM under MK using the MAC scheme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.3 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\overline{EM} = \texttt{01100010 10100100 01000001 11100100 10101101 11110010 10000110 01101011 10101111 11101010 11011010 01010000 10111001 11011010 11000001 00000100 01111011 00101100 10000011 10110011}$$

8.2. Convert MK to a bit string.

$$\overline{MK} = \texttt{00100101 01000111 00101001 00101111 10000010 11011100 01101011 00010111 01110111 11110100 01111101 01100011 10111010 10011101 00011110 10100111 00110010 11011011 11110011 10000110}$$

8.3. Calculate the tag $D = MAC_{\overline{MK}}(\overline{EM})$ using the MAC scheme HMAC-SHA-1-160 with 20 octet keys.

$$D = \texttt{183301B4 14C82DFA 91A58311 369DF0E2 A6F9642C}$$

Output: The ciphertext $C = \overline{R}||EM||D$.

$$C = \texttt{0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B 62A441E4 ADF2866B AFEADA50 B9DAC104 7B2C83B3 183301B4 14C82DFA 91A58311 369DF0E2 A6F9642C}$$

U conveys the encrypted message C to V.

3.2.4 Decryption Operation for V

V decrypts the ciphertext C as follows.

Input: The decryption operation takes the following input.

1. The octet string C which is the ciphertext.

$$C = $$ 0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B
62A441E4 ADF2866B AFEADA50 B9DAC104 7B2C83B3 183301B4
14C82DFA 91A58311 369DF0E2 A6F9642C

2. The optional strings $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: V decrypts the message.

1. Parse C to get $\overline{R}, EM$, and D.

$\overline{R}$ = 0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B
EM = 62A441E4 ADF2866B AFEADA50 B9DAC104 7B2C83B3
D = E5904578 55B8521B 6098F35E 8EB5F0A0 B078E4AD

2. Convert $\overline{R}$ to an elliptic curve point $R = (x_R, y_R)$.

x_R = 04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B
y_R = 03 1FC936D7 3163B858 BBC5326D 77C19839 46405264

3. Validate R using the primitive specified in Section 3.2.2.1 of SEC 1 [1].

   3.1. Verify that $R \neq O$, OK.

   3.2. Verify that R is a point on the curve, OK.

   3.3. Verify that $nR = O$, OK.

4. Derive the shared secret field element z using the standard elliptic curve Diffie Hellman primitive specified in Section 3.3.1 of SEC 1 [1].

   4.1. Compute $P = (x_P, y_P) = d_V \times R$.

x_P = 04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881
y_P = 07 7A8B0052 E8C622CC 3DCC0613 50500262 173EB44E

   4.2. $P \neq O$, OK.

4.3. Set $z = x_P$.

$$z = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881}$$

5. Convert z to an octet string.

$$Z = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881}$$

6. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

6.1. Append $Counter_1 =$ `00000001` to the right of Z.

$$Z_1 = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881 00000001}$$

6.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$

6.3. Append $Counter_2 =$ `00000002` to the right of Z.

$$Z_2 = \texttt{04 99B502FC 8B5BAFB0 F4047E73 1D1F9FD8 CD0D8881 00000002}$$

6.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

6.5. Get $K = Hash_1 || Hash_2$.

$$K = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$
$$\texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

7. Get EK and MK.

7.1. Get EK from K.

$$EK = \texttt{03C62280 C894E103 C680B13C D4B4AE74 0A5EF0C7}$$

7.2. Get MK from K.

$$MK = \texttt{2547292F 82DC6B17 77F47D63 BA9D1EA7 32DBF386}$$

8. Check the tag D on EM under MK using the MAC scheme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.4 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\overline{EM} = \begin{array}{l}\texttt{01100010 10100100 01000001 11100100 10101101 11110010} \\ \texttt{10000110 01101011 10101111 11101010 11011010 01010000} \\ \texttt{10111001 11011010 11000001 00000100 01111011 00101100} \\ \texttt{10000011 10110011}\end{array}$$

8.2. Convert MK to a bit string.

$$\overline{MK} = \begin{array}{l}\texttt{00100101 01000111 00101001 00101111 10000010 11011100} \\ \texttt{01101011 00010111 01110111 11110100 01111101 01100011} \\ \texttt{10111010 10011101 00011110 10100111 00110010 11011011} \\ \texttt{11110011 10000110}\end{array}$$

8.3. Calculate the tag $D' = MAC_{\overline{MK}}(\overline{EM})$ using the MAC scheme HMAC-SHA-1-160 with 20 octet keys.

$$D' = \texttt{183301B4 14C82DFA 91A58311 369DF0E2 A6F9642C}$$

8.4. $D' = D$, OK.

9. Decrypt the octet string EM under EK to produce M using the XOR encryption scheme as specified in Section 3.8.4 of SEC 1 [1].

9.1. Convert EM to a bit string $EM_0EM_1EM_2\dots EM_{159}$.

$$\overline{EM} = \texttt{01100010 10100100 01000001 11100100 10101101 11110010 10000110 01101011 10101111 11101010 11011010 01010000 10111001 11011010 11000001 00000100 01111011 00101100 10000011 10110011}$$

9.2. Convert EK to a bit string $EK_0EK_1EK_2\dots EM_{159}$.

$$\overline{EK} = \texttt{00000011 11000110 00100010 10000000 11001000 10010100 11100001 00000011 11000110 10000000 10110001 00111100 11010100 10110100 10101110 01110100 00001010 01011110 11110000 11000111}$$

9.3. Use XOR to decrypt EM by $\overline{EM} \oplus \overline{EK}$.

$$\overline{M} = \texttt{01100001 01100010 01100011 01100100 01100101 01100110 01100111 01101000 01101001 01101010 01101011 01101100 01101101 01101110 01101111 01110000 01110001 01110010 01110011 01110100}$$

9.4. Convert $\overline{M}$ to an octet string M using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

Output: The message M.

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

M represents the text string “abcdefghijklmnopqrst”.

3.3 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$ and the Cofactor Diffie-Hellman Primitive

This section provides test vectors for ECAES using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$ and the cofactor Diffie-Hellman primitive. U and V use ECAES as follows.

3.3.1 Scheme Setup

V decides to use ECAES with the key derivation function ANSI-X9.63-KDF with SHA-1 specified in Section 3.6 of SEC 1 [1], the MAC scheme HMAC-SHA-1-160 with 20 octet keys, the XOR symmetric encryption scheme, the cofactor Diffie-Hellman primitive, and the elliptic curve domain parameters sect163k1 specified in GEC 1 [2]. V conveys thest decisions to U. U decides to represent elliptic curve points in compressed form.

3.3.2 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.

   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

   $$d_V = 501870566195266176721440888203272826969530834326$$

   1.2. d_V is represented as the octet string $\overline{d_V}$ with:

   $$\overline{d_V} = \texttt{57E8A78E 842BF4AC D5C315AA 0569DB17 03541D96}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

   $$x_V = \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$
   $$y_V = \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7}$$

   As an octet string with point compression we have:

$$\overline{Q_V} = \texttt{0307 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$

Output: The elliptic curve key pair (d_V, Q_V) with:

$$d_V = 501870566195266176721440888203272826969530834326$$
$$Q_V = (\texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C},$$
$$\texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7})$$

V shares Q_V with U in an authentic manner. U should check that Q_V is at least partially valid.

3.3.3 Encryption Operation for U

Suppose U wants to convey the message $M =$ "abcdefghijklmnopqrst" confidentially to V.

Input: The encryption operation takes the following input.

1. The octet string $M =$ `61626364 65666768 696A6B6C 6D6E6F70 71727374` which represents the message "abcdefghijklmnopqrst".

2. The optional strings $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: U encrypts the message M.

1. Select an ephemeral key pair (k, R) using the key pair generation primitive specified in Section 3.2.1 of SEC 1 [1].

   1.1. Randomly or pseudorandomly select an integer k in the interval $[1, n-1]$.

$$k = 936523985789236956265265265235675811949404040044$$

   1.2. Compute $R = (x_R, y_R) = k \times G$.

$$x_R = \texttt{04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B}$$
$$y_R = \texttt{03 1FC936D7 3163B858 BBC5326D 77C19839 46405264}$$

2. Convert the point R to an octet string $\overline{R}$ with point compression using the conversion routine specified in Section 2.3.3 of SEC 1 [1].

$\overline{R}$ = 0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B

3. Compute the shared secret field element z using the cofactor elliptic curve Diffie Hellman primitive specified in Section 3.3.2 of SEC 1 [1].

3.1. Compute $P = (x_P, y_P) = h \times k \times Q_V$.

x_P = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

y_P = 05 4F7BD9DA 2D38F636 B3A74297 88EC21A6 BB61DD31

3.2. $P \neq O$, OK.

3.3. Set $z = x_P$.

z = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

4. Convert z to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

Z = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

5. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

5.1. Append $Counter_1 =$ 00000001 to the right of Z.

Z_1 = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0 00000001

5.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$Hash_1$ = 928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D

5.3. Append $Counter_2 =$ 00000002 to the right of Z.

Z_2 = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0 00000002

5.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540}$$

5.5. Get $K = Hash_1||Hash_2$.

$$K = \texttt{928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D}$$
$$\texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540}$$

6. Get EK and MK.

6.1. Get EK from K.

$$EK = \texttt{928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D}$$

6.2. Get MK from K.

$$MK = \texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540}$$

7. Encrypt the octet string M under EK to produce the ciphertext EM using the XOR encryption scheme specified in Section 3.8.3 of SEC 1 [1].

7.1. Convert M to a bit string $M_0M_1M_2 \ldots M_{159}$.

$$\overline{M} = \texttt{01100001 01100010 01100011 01100100 01100101 01100110}$$
$$\texttt{01100111 01101000 01101001 01101010 01101011 01101100}$$
$$\texttt{01101101 01101110 01101111 01110000 01110001 01110010}$$
$$\texttt{01110011 01110100}$$

7.2. Convert EK to a bit string $EK_0EK_1EK_2 \ldots EK_{159}$.

$$\overline{EK} = \texttt{10010010 10001001 10000110 10100001 10111011 00011010}$$
$$\texttt{01011000 01011010 10011111 10110011 10010101 00100101}$$
$$\texttt{10110011 10011101 00010001 11100001 00001000 10110101}$$
$$\texttt{10001001 00011101}$$

7.3. Use XOR to encrypt M by $\overline{M} \oplus \overline{EK}$.

$$\overline{EM} = \texttt{11110011 11101011 11100101 11000101 11011110 01111100 00111111 00110010 11110110 11011001 11111110 01001001 11011110 11110011 01111110 10010001 01111001 11000111 11111010 01101001}$$

7.4. Convert $\overline{EM}$ to an octet string EM using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$EM = \texttt{F3EBE5C5 DE7C3F32 F6D9FE49 DEF37E91 79C7FA69}$$

8. Compute the tag D on EM under MK using the MAC scheme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.3 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\overline{EM} = \texttt{11110011 11101011 11100101 11000101 11011110 01111100 00111111 00110010 11110110 11011001 11111110 01001001 11011110 11110011 01111110 10010001 01111001 11000111 11111010 01101001}$$

8.2. Convert MK to a bit string.

$$\overline{MK} = \texttt{11010010 10111011 00011101 00011011 01010001 10001101 00010001 01110010 11001001 11010001 10111100 10100011 10011011 10111011 00000011 10010011 01111111 01101111 11000101 01000000}$$

8.3. Calculate the tag $D = MAC_{\overline{MK}}(\overline{EM})$ using the MAC scheme HMAC-SHA-1-160 with 20 octet keys.

$$D = \texttt{AE2DF284 F6E61970 62C9679A 1F6D7C81 79C12EF1}$$

Output: The ciphertext $C = \overline{R}||EM||D$.

$$
\begin{aligned}
C \;=\; & \texttt{0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B} \\
& \texttt{F3EBE5C5 DE7C3F32 F6D9FE49 DEF37E91 79C7FA69 AE2DF284} \\
& \texttt{F6E61970 62C9679A 1F6D7C81 79C12EF1}
\end{aligned}
$$

U conveys the encrypted message C to V.

3.3.4 Decryption Operation for V

V decrypts the ciphertext C as follows.

Input: The decryption operation takes the following input.

1. The octet string C which is the ciphertext.

$$
\begin{aligned}
C \;=\; & \texttt{0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B} \\
& \texttt{F3EBE5C5 DE7C3F32 F6D9FE49 DEF37E91 79C7FA69 AE2DF284} \\
& \texttt{F6E61970 62C9679A 1F6D7C81 79C12EF1}
\end{aligned}
$$

2. The optional strings $SharedInfo_1$ and $SharedInfo_2$ are absent.

Actions: V decrypts the message.

1. Parse C to get $\overline{R}, EM$, and D.

$$
\begin{aligned}
\overline{R} &= \texttt{0304 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B} \\
EM &= \texttt{F3EBE5C5 DE7C3F32 F6D9FE49 DEF37E91 79C7FA69} \\
D &= \texttt{AE2DF284 F6E61970 62C9679A 1F6D7C81 79C12EF1}
\end{aligned}
$$

2. Convert $\overline{R}$ to an elliptic curve point $R = (x_R, y_R)$.

$$
\begin{aligned}
x_R &= \texttt{04 994D2C41 AA30E529 52B0A94E C6511328 C502DA9B} \\
y_R &= \texttt{03 1FC936D7 3163B858 BBC5326D 77C19839 46405264}
\end{aligned}
$$

3. Validate R using the primitive specified in Section 3.2.2.2 of SEC 1 [1].

3.1. Verify that $R \neq \mathcal{O}$, OK.

3.2. Verify that R is a point on the curve, OK.

4. Derive the shared secret field element z using the cofactor elliptic curve Diffie Hellman primitive specified in Section 3.3.2 of SEC 1 [1].

4.1. Compute $P = (x_P, y_P) = h \times d_V \times R$.

x_P = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

y_P = 05 4F7BD9DA 2D38F636 B3A74297 88EC21A6 BB61DD31

4.2. $P \neq \mathcal{O}$, OK.

4.3. Set $z = x_P$.

z = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

5. Convert z to an octet string using the conversion routine specified in Section 2.3.5 of SEC 1 [1].

Z = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0

6. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length $enckeylen + mackeylen = 40$ octets from Z.

6.1. Append $Counter_1 =$ 00000001 to the right of Z.

Z_1 = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0 00000001

6.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$Hash_1$ = 928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D

6.3. Append $Counter_2 =$ 00000002 to the right of Z.

Z_2 = 01 7F645842 C0289769 06DB086B 4C7C455C B3BF53A0 00000002

6.4. Compute $Hash_2 = SHA\text{-}1(Z_2)$.

$$Hash_2 = \texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540}$$

6.5. Get $K = Hash_1 || Hash_2$.

$$\begin{aligned} K = \; & \texttt{928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D} \\ & \texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540} \end{aligned}$$

7. Get EK and MK.

7.1. Get EK from K.

$$EK = \texttt{928986A1 BB1A585A 9FB39525 B39D11E1 08B5891D}$$

7.2. Get MK from K.

$$MK = \texttt{D2BB1D1B 518D1172 C9D1BCA3 9BBB0393 7F6FC540}$$

8. Check the tag D on EM under MK using the MAC scheme HMAC-SHA-1-160 with 20 octet keys as specified in Section 3.7.4 of SEC 1 [1].

8.1. Convert EM to a bit string.

$$\begin{aligned} \overline{EM} = \; & \texttt{11110011 1110101111100101 11000101 11011110 01111100} \\ & \texttt{00111111 00110010 11110110 11011001 11111110 01001001} \\ & \texttt{11011110 11110011 01111110 10010001 01111001 11000111} \\ & \texttt{11111010 01101001} \end{aligned}$$

8.2. Convert MK to a bit string.

$$\begin{aligned} \overline{MK} = \; & \texttt{11010010 10111011 00011101 00011011 01010001 10001101} \\ & \texttt{00010001 01110010 11001001 11010001 10111100 10100011} \\ & \texttt{10011011 10111011 00000011 10010011 01111111 01101111} \\ & \texttt{11000101 01000000} \end{aligned}$$

8.3. Calculate the tag $D' = MAC_{\overline{MK}}(\overline{EM})$ using the MAC scheme HMAC-SHA-1-160 with 20 octet keys.

$$D' = \texttt{AE2DF284 F6E61970 62C9679A 1F6D7C81 79C12EF1}$$

8.4. $D' = D$, OK.

9. Decrypt the octet string EM under EK to produce M using the XOR encryption scheme as specified in Section 3.8.4 of SEC 1 [1].

9.1. Convert EM to a bit string $EM_0 EM_1 EM_2 \ldots EM_{159}$.

$$\begin{aligned}\overline{EM} = \; & \texttt{11110011 11101011 11100101 11000101 11011110 01111100} \\ & \texttt{00111111 00110010 11110110 11011001 11111110 01001001} \\ & \texttt{11011110 11110011 01111110 10010001 01111001 11000111} \\ & \texttt{11111010 01101001}\end{aligned}$$

9.2. Convert EK to a bit string $EK_0 EK_1 EK_2 \ldots EM_{159}$.

$$\begin{aligned}\overline{EK} = \; & \texttt{10010010 10001001 10000110 10100001 10111011 00011010} \\ & \texttt{01011000 01011010 10011111 10110011 10010101 00100101} \\ & \texttt{10110011 10011101 00010001 11100001 00001000 10110101} \\ & \texttt{10001001 00011101}\end{aligned}$$

9.3. Use XOR to decrypt EM by $\overline{EM} \oplus \overline{EK}$.

$$\begin{aligned}\overline{M} = \; & \texttt{01100001 01100010 01100011 01100100 01100101 01100110} \\ & \texttt{01100111 01101000 01101001 01101010 01101011 01101100} \\ & \texttt{01101101 01101110 01101111 01110000 01110001 01110010} \\ & \texttt{01110011 01110100}\end{aligned}$$

9.4. Convert $\overline{M}$ to an octet string M using the conversion routine specified in Section 2.3.1 of SEC 1 [1].

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

Output: The message M.

$$M = \texttt{61626364 65666768 696A6B6C 6D6E6F70 71727374}$$

M represents the text string “abcdefghijklmnopqrst”.

4 Test Vectors for ECDH

This section provides test vectors for ECDH as specified in section 6.1 of SEC 1 [1]. Section 4.1 provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_p$, Section 4.2 provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_{2^m}$ and the standard Diffie-Hellman primitive, and Section 4.3 provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_{2^m}$ and the cofactor Diffie-Hellman primitive.

4.1 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_p$

This section provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_p$. U and V use ECDH as follows.

4.1.1 Scheme Setup

U and V decide to use the key derivation function ANSI-X9.63-KDF with SHA-1 and the elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Note that in this case the cofactor is $h = 1$ so the choice between the standard and cofactor Diffie-Hellman primitives is unnecessary.

4.1.2 Key Deployment for U

U selects a key pair (d_U, Q_U) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: U selects a key pair.

1. Generate an integer d_U.

   1.1. Randomly or pseudorandomly select an integer d_U in the interval $[1, n-1]$.

$$d_U = 971761939728640320549601132085879836204587084162$$

   1.2. Convert d_U to the octet string $\overline{d_U}$.

$$\overline{d_U} = \texttt{AA374FFC 3CE144E6 B0733079 72CB6D57 B2A4E982}$$

2. Calculate $Q_U = (x_U, y_U) = d_U \times G$.

$$x_U = 466448783855397898016055842232266600516272889280$$
$$y_U = 1110706324081757720403272427311003102474457754220$$

As an octet string with point compression, we have:

$$\overline{Q_U} = \texttt{02 51B4496F ECC406ED 0E75A24A 3C032062 51419DC0}$$

Output: The elliptic curve key pair (d_U, Q_U) with:

$$d_U = 971761939728640320549601132085879836204587084162$$
$$Q_U = (466448783855397898016055842232266600516272889280,$$
$$1110706324081757720403272427311003102474457754220)$$

U conveys Q_U to V. V should check that Q_U is valid.

4.1.3 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.

   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

$$d_V = 399525573676508631577122671218044116107572676710$$

   1.2. Convert d_V to the octet string $\overline{d_V}$.

$$\overline{d_V} = \texttt{45FB58A9 2A17AD4B 15101C66 E74F277E 2B460866}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

$$x_V = 420773078745784176406965940076771545932416607676$$
$$y_V = 221937774842090227911893783570676792435918278531$$

As an octet string with point compression we have:

$$\overline{Q_V} = \texttt{03 49B41E0E 9C0369C2 328739D9 0F63D567 07C6E5BC}$$

Output: The key pair (d_V, Q_V).

$$d_V = 399525573676508631577122671218044116107572676710$$
$$Q_V = (420773078745784176406965940076771545932416607676, 221937774842090227911893783570676792435918278531)$$

V conveys Q_V to U. U should check that Q_V is valid.

4.1.4 Key Agreement Operation for U

To agree on keying data, U and V simultaneously perform the key agreement operation. U establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data.

1. Compute the shared secret field element z.

   1.1. Compute $P = (x_P, y_P) = d_U \times Q_V$.

   $$x_P = 1155982782519895915997745984453282631351432623114$$
   $$y_P = 450433377308022757780566139350756069889901357499$$

   1.2. Verify that $P \neq O$, OK.

1.3. Set $z = x_P$.

$$z = 1155982782519895915997745984453282631351432623114$$

1.4. Convert z to an octet string.

$$Z = \texttt{CA7C0F8C 3FFA87A9 6E1B74AC 8E6AF594 347BB40A}$$

2. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

2.1. Append $Counter_1 =$ `00000001` to the right of Z.

$$Z_1 = \texttt{CA7C0F8C 3FFA87A9 6E1B74AC 8E6AF594 347BB40A 00000001}$$

2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2}$$

2.3. Get $K = Hash_1$.

$$K = \texttt{744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2}$$

Output: The keying data K.

$$K = \texttt{744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2}$$

4.1.5 Key Agreement Operation for V

To agree on keying data, U and V simultaneously perform the key agreement operation. V establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: V establishes keying data.

1. Compute the shared secret field element z.

   1.1. Compute $P = (x_P, y_P) = d_V \times Q_U$.

$$x_P = 1155982782519895915997745984453282631351432623114$$
$$y_P = 450433377308022757780566139350756069889901357499$$

   1.2. $P \neq O$, OK.

   1.3. Set $z = x_P$.

$$z = 1155982782519895915997745984453282631351432623114$$

   1.4. Convert z to an octet string.

Z = `CA7C0F8C 3FFA87A9 6E1B74AC 8E6AF594 347BB40A`

2. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   2.1. Append $Counter_1$ = `00000001` to the right of Z.

Z_1 = `CA7C0F8C 3FFA87A9 6E1B74AC 8E6AF594 347BB40A 00000001`

   2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$Hash_1$ = `744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2`

   2.3. Get $K = Hash_1$.

K = `744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2`

Output: The keying data K.

K = `744AB703 F5BC082E 59185F6D 049D2D36 7DB245C2`

4.2 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$ and the Standard Diffie-Hellman Primitive

This section provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$ and the standard Diffie-Hellman primitive. U and V use ECDH as follows.

4.2.1 Scheme Setup

U and V decide to use the key derivation function ANSI-X9.63-KDF with SHA-1, the standard elliptic curve Diffie-Hellman primitive, and the elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

4.2.2 Key Deployment for U

U selects a key pair (d_U, Q_U) as follows using the key generation primitive specified Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: U selects a key pair.

1. Generate an integer d_U.

   1.1. Randomly or pseudorandomly select an integer d_U in the interval $[1, n-1]$.

$$d_U = 5321230001203043918714616464614664646674949479949$$

   1.2. Convert d_U to the octet string $\overline{d_U}$.

$$\overline{d_U} = \texttt{03 A41434AA 99C2EF40 C8495B2E D9739CB2 155A1E0D}$$

2. Calculate $Q_U = (x_U, y_U) = d_U \times G$.

$$x_U = \texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$
$$y_U = \texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776}$$

   As an octet string with point compression we have:

$$\overline{Q_U} = \texttt{0303 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$

Output: The elliptic curve key pair (d_U, Q_U) with:

$$d_U = 5321230001203043918714616464614664646674949479949$$
$$Q_U = (\texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC},$$
$$\texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776})$$

U conveys Q_U to V. V should check that Q_U is valid.

4.2.3 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.

   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

   $$d_V = 501870566195266176721440888203272826969530834326$$

   1.2. Convert d_V to an octet string $\overline{d_V}$.

   $$\overline{d_V} = \texttt{57E8A78E 842BF4AC D5C315AA 0569DB17 03541D96}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

   $$x_V = \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$
   $$y_V = \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7}$$

   As an octet string with point compression we have:

   $$\overline{Q_V} = \texttt{0307 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$

Output: The key pair (d_V, Q_V).

$$
\begin{aligned}
d_V &= 501870566195266176721440888203272826969530834326 \\
Q_V &= (\quad \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}, \\
&\qquad \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7})
\end{aligned}
$$

V conveys Q_V to U. U should check that Q_V is valid.

4.2.4 Key Agreement Operation for U

To agree on keying data, U and V simultaneously perform the key agreement operation. U establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data.

1. Compute the shared secret field element z.

   1.1. Compute $P = (x_P, y_P) = d_U \times Q_V$.

$$
\begin{aligned}
x_P &= \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7} \\
y_P &= \texttt{01 9A2DC185 889E0FBA 4D81DE92 CED18878 DDCC5AC2}
\end{aligned}
$$

   1.2. $P \neq O$, OK.

   1.3. Set $z = x_P$.

$$
z = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7}
$$

   1.4. Convert z to an octet string.

$$
Z = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7}
$$

2. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

2.1. Append $Counter_1 = $ 00000001 to the right of Z.

$$Z_1 = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7 00000001}$$

2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

2.3. Get $K = Hash_1$.

$$K = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

Output: The keying data K.

$$K = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

4.2.5 Key Agreement Operation for V

To agree on keying data, U and V simultaneously perform the key agreement operation. V establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data.

1. Compute the shared secret field element z.

1.1. Compute $P = (x_P, y_P) = d_V \times Q_U$.

$$x_P = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7}$$
$$y_P = \texttt{01 9A2DC185 889E0FBA 4D81DE92 CED18878 DDCC5AC2}$$

1.2. $P \neq O$, OK.

1.3. Set $z = x_P$.

$$z = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7}$$

1.4. Convert z to an octet string.

$$Z = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7}$$

2. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

2.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

$$Z_1 = \texttt{03 57C3DCD1 DF3E27BD 8885170E E4975B50 81DA7FA7 00000001}$$

2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

2.3. Get $K = Hash_1$.

$$K = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

Output: The keying data K.

$$K = \texttt{6655A9C8 F9E59314 9DB24C91 CE621641 035C9282}$$

4.3 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$ and the Cofactor Diffie-Hellman Primitive

This section provides test vectors for ECDH using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$ and the cofactor Diffie-Hellman primitive. U and V use ECDH as follows.

4.3.1 Scheme Setup

U and V decide to use the key derivation function ANSI-X9.63-KDF with SHA-1, the cofactor elliptic curve Diffie-Hellman primitive, and the elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

4.3.2 Key Deployment for U

U selects a key pair (d_U, Q_U) as follows using the key generation primitive specified Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: U selects a key pair.

1. Generate an integer d_U.

   1.1. Randomly or pseudorandomly select an integer d_U in the interval $[1, n-1]$.

   $$d_U = 5321230001203043918714616464614664646674949479949$$

   1.2. Convert d_U to the octet string $\overline{d_U}$.

   $$\overline{d_U} = \texttt{03 A41434AA 99C2EF40 C8495B2E D9739CB2 155A1E0D}$$

2. Calculate $Q_U = (x_U, y_U) = d_U \times G$.

   $$x_U = \texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$
   $$y_U = \texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776}$$

   As an octet string with point compression we have:

   $$\overline{Q_U} = \texttt{0303 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$

Output: The elliptic curve key pair (d_U, Q_U) with:

$$d_U = 5321230001203043918714616464614664646674949479949$$
$$Q_U = (\texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC},$$
$$\texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776})$$

U conveys Q_U to V. V should check that Q_U is at least partially valid.

4.3.3 Key Deployment for V

V selects a key pair (d_V, Q_V) as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: V selects a key pair.

1. Generate an integer d_V.
   1.1. Randomly or pseudorandomly select an integer d_V in the interval $[1, n-1]$.

   $$d_V = 501870566195266176721440888203272826969530834326$$

   1.2. Convert d_V to an octet string $\overline{d_V}$.

   $$\overline{d_V} = \texttt{57E8A78E 842BF4AC D5C315AA 0569DB17 03541D96}$$

2. Calculate $Q_V = (x_V, y_V) = d_V \times G$.

   $$x_V = \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$
   $$y_V = \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7}$$

   As an octet string with point compression we have:

   $$\overline{Q_V} = \texttt{0307 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$

Output: The key pair (d_V, Q_V).

$$d_V = 501870566195266176721440888203272826969530834326$$
$$Q_V = (\texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}, \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7})$$

V conveys Q_V to U. U should check that Q_V is at least partially valid.

4.3.4 Key Agreement Operation for U

To agree on keying data, U and V simultaneously perform the key agreement operation. U establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data.

1. Compute the shared secret field element z.

   1.1. Compute $P = (x_P, y_P) = h \times d_U \times Q_V$.

   x_P = 04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D
   y_P = 00 6C1EBD49 57115DE5 F033D926 7F35875A 44AF87E9

   1.2. $P \neq O$, OK.

   1.3. Set $z = x_P$.

   z = 04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D

   1.4. Convert z to an octet string.

   Z = 04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D

2. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   2.1. Append $Counter_1 = 00000001$ to the right of Z.

   Z_1 = 04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D 00000001

2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

2.3. Get $K = Hash_1$.

$$K = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

Output: The keying data K.

$$K = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

4.3.5 Key Agreement Operation for V

To agree on keying data, U and V simultaneously perform the key agreement operation. V establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: V establishes keying data.

1. Compute the shared secret field element z.

1.1. Compute $P = (x_P, y_P) = h \times d_V \times Q_U$.

$$x_P = \texttt{04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D}$$
$$y_P = \texttt{00 6C1EBD49 57115DE5 F033D926 7F35875A 44AF87E9}$$

1.2. $P \neq O$, OK.

1.3. Set $z = x_P$.

$$z = \texttt{04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D}$$

1.4. Convert z to an octet string.

$$Z = \texttt{04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D}$$

2. Use the key derivation function ANSI-X9.53-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

2.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

$$Z_1 = \texttt{04 CB89474B 33A518E1 C3CD11BE B6E2B0CF 48BEE64D 00000001}$$

2.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

2.3. Get $K = Hash_1$.

$$K = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

Output: The keying data K.

$$K = \texttt{59798528 083F50B0 7528353C DA99D0E4 60A7229D}$$

5 Test Vectors for ECMQV

This section provides test vectors for ECMQV as specified in Section 6.2 of SEC 1 [1]. Section 5.1 provides test vectors for ECMQV using elliptic curve domain parameters over $\mathbb{F}_p$, and Section 5.2 provides test vectors for ECMQV using elliptic curve domain parameters over $\mathbb{F}_{2^m}$.

5.1 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_p$

This section provides test vectors for ECMQV using elliptic curve domain parameters over $\mathbb{F}_p$. U and V use ECMQV as follows.

5.1.1 Scheme Setup

U and V decide to use the key derivation function ANSI-X9.63-KDF with SHA-1 and the elliptic curve domain parameters secp160r1 specified in GEC 1 [2].

5.1.2 Key Deployment for U

U selects two key pairs $(d_{1,U}, Q_{1,U})$ and $(d_{2,U}, Q_{2,U})$ as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: U selects two key pairs.

1. Generate an integer $d_{1,U}$.

    1.1. Randomly or pseudorandomly select an integer $d_{1,U}$ in the interval $[1, n-1]$.

    $$d_{1,U} = 971761939728640320549601132085879836204587084162$$

    1.2. Convert $d_{1,U}$ to the octet string $\overline{d_{1,U}}$.

    $$\overline{d_{1,U}} = \texttt{AA374FFC 3CE144E6 B0733079 72CB6D57 B2A4E982}$$

2. Calculate $Q_{1,U} = (x_{1,U}, y_{1,U}) = d_{1,U} \times G$.

$$x_{1,U} = 466448783855397898016055842232266600516272889280$$
$$y_{1,U} = 1110706324081757720403272427311003102474457754220$$

As an octet string with point compression we have:

$$\overline{Q_{1,U}} = \texttt{02 51B4496F ECC406ED 0E75A24A 3C032062 51419DC0}$$

3. Generate an integer $d_{2,U}$.

   3.1. Randomly or pseudorandomly select an integer $d_{2,U}$ in the interval $[1, n-1]$.

$$d_{2,U} = 117720748206090884214100397070943062470184499100$$

   3.2. Convert $d_{2,U}$ to the octet string $\overline{d_{2,U}}$.

$$\overline{d_{2,U}} = \texttt{149EC7EA 3A220A88 7619B3F9 E5B4CA51 C7D1779C}$$

4. Calculate $Q_{2,U} = (x_{2,U}, y_{2,U}) = d_{2,U} \times G$.

$$x_{2,U} = 1242349848876241038961169594145217616154763512351$$
$$y_{2,U} = 1228723083615049968259530566733073401525145323751$$

As an octet string with point compression we have:

$$\overline{Q_{2,U}} = \texttt{03 D99CE4D8 BF52FA20 BD21A962 C6556B0F 71F4CA1F}$$

Output: The two elliptic curve key pairs $(d_{1,U}, Q_{1,U})$ and $(d_{2,U}, Q_{2,U})$ with:

$$d_{1,U} = 971761939728640320549601132085879836204587084162$$
$$Q_{1,U} = (466448783855397898016055842232266600516272889280, 1110706324081757720403272427311003102474457754220)$$

$$d_{2,U} = 117720748206090884214100397070943062470184499100$$
$$Q_{2,U} = (1242349848876241038961169594145217616154763512351, 1228723083615049968259530566733073401525145323751)$$

U shares $Q_{1,U}$ with V in an authentic manner. V should check that $Q_{1,U}$ is valid. In addition, U shares $Q_{2,U}$ with V. V should check that $Q_{2,U}$ is at least partially valid.

5.1.3 Key Deployment for V

V selects two key pairs $(d_{1,V}, Q_{1,V})$ and $(d_{2,V}, Q_{2,V})$ as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters secp160r1 as specified in GEC 1 [2].

Actions: V selects two key pairs.

1. Generate an integer $d_{1,V}$.

   1.1. Randomly or pseudorandomly select an integer $d_{1,V}$ in the interval $[1, n-1]$.

   $$d_{1,V} = 399525573676508631577122671218044116107572676710$$

   1.2. Convert $d_{1,V}$ to the octet string $\overline{d_{1,V}}$.

   $$\overline{d_{1,V}} = \texttt{45FB58A9 2A17AD4B 15101C66 E74F277E 2B460866}$$

2. Calculate $Q_{1,V} = (x_{1,V}, y_{1,V}) = d_{1,V} \times G$.

   $$x_{1,V} = 420773078745784176406965940076771545932416607676$$
   $$y_{1,V} = 221937774842090227911893783570676792435918278531$$

   As an octet string with point compression we have:

   $$\overline{Q_{1,V}} = \texttt{03 49B41E0E 9C0369C2 328739D9 0F63D567 07C6E5BC}$$

3. Generate an integer $d_{2,V}$.

   3.1. Randomly or pseudorandomly select an integer $d_{2,V}$ in the interval $[1, n-1]$.

   $$d_{2,V} = 141325380784931851783969312377642205317371311134$$

   3.2. Convert $d_{2,V}$ to an octet string $\overline{d_{2,V}}$.

   $$\overline{d_{2,V}} = \texttt{18C13FCE D9EADF88 4F7C595C 8CB565DE FD0CB41E}$$

4. Calculate $Q_{2,V} = (x_{2,V}, y_{2,V}) = d_{2,V} \times G$.

$$x_{2,V} = 641868187219485959973483930084949222543277290421$$
$$y_{2,V} = 560813476551307469487939594456722559518188737232$$

As an octet string with point compression we have:

$$\overline{Q_{2,V}} = \texttt{02 706E5D6E 1F640C6E 9C804E75 DBC14521 B1E5F3B5}$$

Output: The two elliptic curve key pairs $(d_{1,V}, Q_{1,V})$ and $(d_{2,V}, Q_{2,V})$ with:

$$d_{1,V} = 399525573676508631577122671218044116107572676710$$
$$Q_{1,V} = (420773078745784176406965940076771545932416607676, 221937774842090227911893783570676792435918278531)$$

$$d_{2,V} = 141325380784931851783969312377642205317371311134$$
$$Q_{2,V} = (641868187219485959973483930084949222543277290421, 560813476551307469487939594456722559518188737232)$$

V shares $Q_{1,V}$ with U in an authentic manner. U should check that $Q_{1,V}$ is valid. In addition, V shares $Q_{2,V}$ with U. U should check that $Q_{2,V}$ is at least partially valid.

5.1.4 Key Agreement Operation for U

To agree on keying data, U and V simultaneously perform the key agreement operation. U establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data as follows:

1. Set $t = \lceil (\log_2 n)/2 \rceil = 81$.

2. Compute $\overline{\overline{Q_{2,U}}}$ using $Q_{2,U}$.

2.1. Parse $Q_{2,U}$ to get $x_{2,U}$.

$$x_{2,U} = 1242349848876241038961169594145217616154763512351$$

2.2. Convert $x_{2,U}$ to an integer x using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$x = 1242349848876241038961169594145217616154763512351$$

2.3. Calculate $\bar{x} \equiv x \pmod{2^t}$.

$$\bar{x} = 2008827827585529362565663$$

2.4. Calculate $\overline{\overline{Q_{2,U}}} = \bar{x} + 2^t$.

$$\overline{\overline{Q_{2,U}}} = 4426679466814787711978015$$

3. Compute the integer $s \equiv d_{2,U} + \overline{\overline{Q_{2,U}}} \cdot d_{1,U} \pmod{n}$.

$$s = 4856188333885434143076888914846925882659664798 53$$

4. Compute $\overline{\overline{Q_{2,V}}}$ using $Q_{2,V}$.

4.1. Parse $Q_{2,V}$ to get $x_{2,V}$.

$$x_{2,V} = 6418681872194859599734839300849492225432772904 21$$

4.2. Convert $x_{2,V}$ to an integer x' using the conversion routine specified in section 2.3.9 of SEC 1 [1].

$$x' = 641868187219485959973483930084949222543277290421$$

4.3. Calculate $\bar{x} \equiv x' \pmod{2^t}$.

$$\bar{x}' = 3705186897342321764566 29$$

4.4. Calculate $\overline{\overline{Q_{2,V}}} = \bar{x}' + 2^t$.

$$\overline{\overline{Q_{2,V}}} = 2788370328963490525868981$$

5. Compute the elliptic curve point $P = (x_P, y_P) = hs \times (Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V})$.

5.1. Compute $\overline{\overline{Q_{2,V}}} Q_{1,V} = (x_{V_{T_1}}, y_{V_{T_1}})$.

$$\begin{aligned} x_{V_{T_1}} &= 5325554128843478757334767211728065922253228 28515 \\ y_{V_{T_1}} &= 9554875927081951366988478046520292871054049 0475 \end{aligned}$$

5.2. Compute $Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V} = (x_{V_{T_2}}, y_{V_{T_2}})$.

$$\begin{aligned} x_{V_{T_2}} &= 3866096511636286833268069366387515123433707 8882 \\ y_{V_{T_2}} &= 3194116279917153813944745945314291974365098 74524 \end{aligned}$$

5.3. Compute $P = (x_P, y_P) = hs \times (Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V})$.

$$\begin{aligned} x_P &= 5161582225996969826906606488016825844322699 85196 \\ y_P &= 4118883524544453658834413533274541641855450 84440 \end{aligned}$$

6. Verify that $P \neq O$, OK.

7. Set $z = x_P$.

$$z = 5161582225996969826906606488016825844322699 85196$$

8. Convert z to an octet string.

$$Z = \texttt{5A6955CE FDB4E432 55FB7FCF 718611E4 DF8E05AC}$$

9. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   9.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

   $$Z_1 = \texttt{5A6955CE FDB4E432 55FB7FCF 718611E4 DF8E05AC 00000001}$$

   9.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

   $$Hash_1 = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

   9.3. Get $K = Hash_1$.

   $$K = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

Output: The keying data K.

$$K = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

5.1.5 Key Agreement Operation for V

To agree on keying data, U and V simultaneously perform the key agreement operation. V establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: V establishes keying data as follows:

1. Set $t = \lceil (\log_2 n)/2 \rceil = 81$.
2. Compute $\overline{\overline{Q_{2,V}}}$ using $Q_{2,V}$.

   2.1. Parse $Q_{2,V}$ to get $x_{2,V}$.

   $$x_{2,V} = 641868187219485959973483930084949222543277290421$$

2.2. Convert $x_{2,V}$ to an integer x using the conversion routine sepcified in Section 2.3.9 of SEC 1 [1].

$$x = 641868187219485959973483930084949222543277290421$$

2.3. Calculate $\overline{x} \equiv x \pmod{2^t}$.

$$\overline{x} = 370518689734232176456629$$

2.4. Calculate $\overline{\overline{Q_{2,V}}} = \overline{x} + 2^t$.

$$\overline{\overline{Q_{2,V}}} = 278837032896349052586898 1$$

3. Compute the integer $s \equiv d_{2,V} + \overline{\overline{Q_{2,V}}} \cdot d_{1,V} \pmod{n}$.

$$s = 933423399729221564875924570036034619821359046776$$

4. Compute $\overline{\overline{Q_{2,U}}}$ using $Q_{2,U}$.

4.1. Parse $Q_{2,U}$ to get $x_{2,U}$.

$$x_{2,U} = 1242349848876241038961169594145217616154763512351$$

4.2. Convert $x_{2,U}$ to an integer x' using the conversion routine as specified in Section 2.3.9 of SEC 1 [1].

$$x' = 1242349848876241038961169594145217616154763512351$$

4.3. Calculate $\overline{x}' \equiv x' \pmod{2^t}$.

$$\overline{x}' = 200882782758552936256566 3$$

4.4. Calculate $\overline{\overline{Q_{2,U}}} = \overline{x}' + 2^t$.

$$\overline{\overline{Q_{2,U}}} = 442667946681478771197801 5$$

5. Compute the elliptic curve point $P = (x_P, y_P) = hs \times (Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U})$.

   5.1. Compute $\overline{\overline{Q_{2,U}}} Q_{1,U} = (x_{U_{T_1}}, y_{U_{T_1}})$.

$$x_{U_{T_1}} = 2273943317604589870978762695965870752260766510077$$
$$y_{U_{T_1}} = 1327622488808903866689109299167870822826627972702$$

   5.2. Compute $Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U} = (x_{U_{T_2}}, y_{U_{T_2}})$.

$$x_{U_{T_2}} = 790217483310858520035869091200113478733837067585$$
$$y_{U_{T_2}} = 1069201588477192429889197466620996739557658436282$$

   5.3. Compute $P = (x_P, y_P) = hs \times (Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U})$.

$$x_P = 516158222599696982690660648801682584432269985196$$
$$y_P = 411888352454445365883441353327454164185545084440$$

6. Verify that $P \neq O$, OK.

7. Set $z = x_P$.

$$z = 516158222599696982690660648801682584432269985196$$

8. Convert z to an octet string.

$$Z = \texttt{5A6955CE FDB4E432 55FB7FCF 718611E4 DF8E05AC}$$

9. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   9.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

$$Z_1 = \texttt{5A6955CE FDB4E432 55FB7FCF 718611E4 DF8E05AC 00000001}$$

9.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

9.3. Get $K = Hash_1$.

$$K = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

Output: The keying data K.

$$K = \texttt{C06763F8 C3D2452C 1CC5D29B D61918FB 485063F6}$$

5.2 Example Using Elliptic Curve Domain Parameters over $\mathbb{F}_{2^{163}}$

This section provides test vectors for ECMQV using elliptic curve domain parameters over $\mathbb{F}_{2^{163}}$. U and V use ECMQV as follows.

5.2.1 Scheme Setup

U and V decide to use the key derivation function ANSI-X9.63-KDF with SHA-1 and the elliptic curve domain parameters sect163r1 specified in GEC 1 [2].

5.2.2 Key Deployment for U

U selects two key pairs $(d_{1,U}, Q_{1,U})$ and $(d_{2,U}, Q_{2,U})$ as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: U selects two key pairs.

1. Generate an integer $d_{1,U}$.

    1.1. Randomly or pseudorandomly select an integer $d_{1,U}$ in the interval $[1, n-1]$.

$$d_{1,U} = 5321230001203043918714616464614664646674949479949$$

1.2. Convert $d_{1,U}$ to the octet string $\overline{d_{1,U}}$.

$$\overline{d_{1,U}} = \texttt{03 A41434AA 99C2EF40 C8495B2E D9739CB2 155A1E0D}$$

2. Calculate $Q_{1,U} = (x_{1,U}, y_{1,U}) = d_{1,U} \times G$.

$$x_{1,U} = \texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$
$$y_{1,U} = \texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776}$$

As an octet string with point compression we have:

$$\overline{Q_{1,U}} = \texttt{0303 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}$$

3. Generate an integer $d_{2,U}$.

3.1. Randomly or pseudorandomly select an integer $d_{2,U}$ in the interval $[1, n-1]$.

$$d_{2,U} = 4657215681533189829603817817038616871919531441490$$

3.2. Convert $d_{2,U}$ to an octet string $\overline{d_{2,U}}$.

$$\overline{d_{2,U}} = \texttt{03 2FC4C61A 8211E6A7 C4B8B0C0 3CF35F7C F20DBD52}$$

4. Calculate $Q_{2,U} = (x_{2,U}, y_{2,U}) = d_{2,U} \times G$.

$$x_{2,U} = \texttt{01 5198E74B C2F1E5C9 A62B8024 8DF0D62B 9ADF8429}$$
$$y_{2,U} = \texttt{04 6B206B42 77356574 9F123911 C50992F4 1E5CB048}$$

As an octet string with point compression we have:

$$\overline{Q_{2,U}} = \texttt{0201 5198E74B C2F1E5C9 A62B8024 8DF0D62B 9ADF8429}$$

Output: The two elliptic curve key pairs $(d_{1,U}, Q_{1,U})$ and $(d_{2,U}, Q_{2,U})$ with.

$$d_{1,U} = 5321230001203043918714616464614664646674949479949$$
$$Q_{1,U} = (\texttt{03 7D529FA3 7E42195F 10111127 FFB2BB38 644806BC}, \texttt{04 47026EEE 8B34157F 3EB51BE5 185D2BE0 249ED776})$$

$$d_{2,U} = 4657215681533189829603817817038616871919531441490$$
$$Q_{2,U} = (\texttt{01 5198E74B C2F1E5C9 A62B8024 8DF0D62B 9ADF8429}, \texttt{04 6B206B42 77356574 9F123911 C50992F4 1E5CB048})$$

U shares $Q_{1,U}$ with V in an authentic manner. V should check that $Q_{1,U}$ is valid. In addition, U shares $Q_{2,U}$ with V. V should check that $Q_{2,U}$ is at least partially valid.

5.2.3 Key Deployment for V

V selects two key pairs $(d_{1,V}, Q_{1,V})$ and $(d_{2,V}, Q_{2,V})$ as follows using the key generation primitive specified in Section 3.2.1 of SEC 1 [1].

Input: The elliptic curve domain parameters sect163k1 as specified in GEC 1 [2].

Actions: V selects two key pairs.

1. Generate an integer $d_{1,V}$.

   1.1. Randomly or pseudorandomly select an integer $d_{1,V}$ in the interval $[1, n-1]$.

   $$d_{1,V} = 5018705661952661767214408882032728269695308343226$$

   1.2. Convert $d_{1,V}$ to an octet string $\overline{d_{1,V}}$.

   $$\overline{d_{1,V}} = \texttt{57E8A78E 842BF4AC D5C315AA 0569DB17 03541D96}$$

2. Calculate $Q_{1,V} = (x_{1,V}, y_{1,V}) = d_{1,V} \times G$.

$$x_{1,V} = \texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$
$$y_{1,V} = \texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7}$$

As an octet string with point compression we have:

$$\overline{Q_{1,V}} = \texttt{0307 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C}$$

3. Generate an integer $d_{2,V}$.

   3.1. Randomly or pseudorandomly select an integer $d_{2,V}$ in the interval $[1, n-1]$.

$$d_{2,V} = 4002572202383399431900003559390459361505597843791$$

   3.2. Convert $d_{2,V}$ to an octet string $\overline{d_{2,V}}$.

$$\overline{d_{2,V}} = \texttt{02 BD198B83 A667A8D9 08EA1E6F 90FD5C6D 695DE94F}$$

4. Calculate $Q_{2,V} = (x_{2,V}, y_{2,V}) = d_{2,V} \times G$.

$$x_{2,V} = \texttt{06 7E3AEA35 10D69E8E DD19CB2A 703DDC6C F5E56E32}$$
$$y_{2,V} = \texttt{06 76C1358A 4EEA8050 564C6E82 8385DCE1 427152EB}$$

As an octet string with point compression we have:

$$\overline{Q_{2,V}} = \texttt{0306 7E3AEA35 10D69E8E DD19CB2A 703DDC6C F5E56E32}$$

Output: The two elliptic curve key pairs $(d_{1,V}, Q_{1,V})$ and $(d_{2,V}, Q_{2,V})$ with:

$$d_{1,V} = 5018705661952661767214408882032728269695308343326$$
$$Q_{1,V} = (\texttt{07 2783FAAB 9549002B 4F13140B 88132D1C 75B3886C},$$
$$\texttt{05 A976794E A79A4DE2 6E2E1941 8F097942 C08641C7})$$

$$d_{2,V} = 4002572202383399431900003559390459361505597843791$$
$$Q_{2,V} = (\texttt{06 7E3AEA35 10D69E8E DD19CB2A 703DDC6C F5E56E32},$$
$$\texttt{06 76C1358A 4EEA8050 564C6E82 8385DCE1 427152EB})$$

V shares $Q_{1,V}$ with U in an authentic manner. U should check that $Q_{1,V}$ is valid. In addition, V shares $Q_{2,V}$ with U. U should check that $Q_{2,V}$ is at least partially valid.

5.2.4 Key Agreement Operation for U

To agree on keying data, U and V simultaneously perform the key agreement operation. U establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.
2. The optional string $SharedInfo$ is absent.

Actions: U establishes keying data as follows:

1. Set $t = \lceil (\log_2 n)/2 \rceil = 82$.
2. Compute $\overline{\overline{Q_{2,U}}}$ using $Q_{2,U}$.
   2.1. Parse $Q_{2,U}$ to get $x_{2,U}$.

   $$x_{2,U} \quad = \quad \texttt{01 5198E74B C2F1E5C9 A62B8024 8DF0D62B 9ADF8429}$$

   2.2. Convert $x_{2,U}$ to an integer x using the conversion routing specified in Section 2.3.9 of SEC 1 [1].

   $$x \quad = \quad 1927339751756164565444710620848523211420513305641$$

   2.3. Calculate $\overline{x} \equiv x \pmod{2^t}$.

   $$\overline{x} \quad = \quad 4231914679348092184003625$$

   2.4. Calculate $\overline{\overline{Q_{2,U}}} = \overline{x} + 2^t$.

   $$\overline{\overline{Q_{2,U}}} = 9067617957806608882828329$$

3. Compute the integer: $s \equiv d_{2,U} + \overline{\overline{Q_{2,U}}} \cdot d_{1,U} \pmod{n}$.

$$s \quad = \quad 5558461394684286933414284105086780726014791562704$$

4. Compute $\overline{\overline{Q_{2,V}}}$ using $Q_{2,V}$.

4.1. Parse $Q_{2,V}$ to get $x_{2,V}$.

$$x_{2,V} = \texttt{06 7E3AEA35 10D69E8E DD19CB2A 703DDC6C F5E56E32}$$

4.2. Convert $x_{2,V}$ to an integer x' using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$x' = 9489656506662991559524977000919079181712074567218$$

4.3. Calculate $\overline{x}' \equiv x' \pmod{2^t}$.

$$\overline{x}' = 2168349066751129321565746$$

4.4. Calculate $\overline{\overline{Q_{2,V}}} = \overline{x}' + 2^t$.

$$\overline{\overline{Q_{2,V}}} = 7004052345209646020390450$$

5. Compute the elliptic curve point $P = (x_P, y_P) = hs \times (Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V})$.

5.1. Compute $\overline{\overline{Q_{2,V}}} Q_{1,V} = (x_{V_{T_1}}, y_{V_{T_1}})$.

$$x_{V_{T_1}} = \texttt{05 12570FF3 BF8D099C 2E1DD7CC 18B7F046 68111D51}$$
$$y_{V_{T_1}} = \texttt{05 707BAF0C B9DF3A89 C3BF5CB6 2670A91A 05B0277D}$$

5.2. Compute $Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V} = (x_{V_{T_2}}, y_{V_{T_2}})$.

$$x_{V_{T_2}} = \texttt{07 4B9CB7E5 57683220 3E410F19 D34D0A34 044ADEFD}$$
$$y_{V_{T_2}} = \texttt{06 8D82B90C 77031F42 CE575B5D 9DE8CBA5 799DB3DD}$$

5.3. Compute $P = (x_P, y_P) = hs \times (Q_{2,V} + \overline{\overline{Q_{2,V}}} Q_{1,V})$.

$$x_P = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503}$$
$$y_P = \texttt{03 3D2E58C1 B3A46B6C EC8CE8BB 1415D368 D8F47C6E}$$

6. Verify that $P \neq O$, OK.

7. Set $z = x_P$.

$$z = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503}$$

8. Convert z to an octet string.

$$Z = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503}$$

9. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   9.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

$$Z_1 = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503 00000001}$$

   9.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

$$Hash_1 = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

   9.3. Get $K = Hash_1$.

$$K = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

Output: The keying data K.

$$K = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

5.2.5 Key Agreement Operation for V

To agree on keying data, U and V simultaneously perform the key agreement operation. V establishes keying data as follows.

Input: The key agreement operation takes the following input.

1. The integer $keydatalen = 20$ which is the number of octets of keying data to be produced.

2. The optional string $SharedInfo$ is absent.

Actions: V establishes keying data as follows:

1. Set $t = \lceil(\log_2 n)/2\rceil = 82$.

2. Compute $\overline{\overline{Q_{2,V}}}$ using $Q_{2,V}$.

   2.1. Parse $Q_{2,V}$ to get $x_{2,V}$.

   $$x_{2,V} = \texttt{06 7E3AEA35 10D69E8E DD19CB2A 703DDC6C F5E56E32}$$

   2.2. Convert $x_{2,V}$ to an integer x using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

   $$x = 9489656506662991559524977000919079181712074567218$$

   2.3. Calculate $\overline{x} \equiv x \pmod{2^t}$.

   $$\overline{x} = 2168349066751129321565746$$

   2.4. Calculate $\overline{\overline{Q_{2,V}}} = \overline{x} + 2^t$.

   $$\overline{\overline{Q_{2,V}}} = 7004052345209646020390450$$

3. Compute the integer: $s \equiv d_{2,V} + \overline{\overline{Q_{2,V}}} \cdot d_{1,V} \pmod{n}$.

   $$s = 11667311984256211152855016797034321425185620485850$$

4. Compute $\overline{\overline{Q_{2,U}}}$ using $Q_{2,U}$.

   4.1. Parse $Q_{2,U}$ to get $x_{2,U}$.

   $$x_{2,U} = \texttt{01 5198E74B C2F1E5C9 A62B8024 8DF0D62B 9ADF8429}$$

4.2. Convert $x_{2,U}$ to an integer x' using the conversion routine specified in Section 2.3.9 of SEC 1 [1].

$$x' = 1927339751756164565444710620848523211420513305641$$

4.3. Calculate $\overline{x}' \equiv x' \pmod{2^t}$.

$$\overline{x}' = 423191467934809218400362 5$$

4.4. Calculate $\overline{\overline{Q_{2,U}}} = \overline{x}' + 2^t$.

$$\overline{\overline{Q_{2,U}}} = 906761795780660888282832 9$$

5. Compute the elliptic curve point $P = (x_P, y_P) = hs \times (Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U})$.

5.1. Compute $\overline{\overline{Q_{2,U}}} Q_{1,U} = (x_{U_{T_1}}, y_{U_{T_1}})$.

$x_{U_{T_1}}$ = 04 20A11505 4CAC5002 08EB97FD FE5108E3 A3583472

$y_{U_{T_1}}$ = 02 0A5C1285 A25C9A0C 998FFAD3 43B2D4AB EFC22DFA

5.2. Compute $Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U} = (x_{U_{T_2}}, y_{U_{T_2}})$.

$x_{U_{T_2}}$ = 07 B798B831 2F361739 12601591 C88162E7 39934166

$y_{U_{T_2}}$ = 00 E19C2CA0 CF7EB554 2E98C197 F8AAACA5 275553A5

5.3. Compute $P = (x_P, y_P) = hs \times (Q_{2,U} + \overline{\overline{Q_{2,U}}} Q_{1,U})$.

x_P = 03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503

y_P = 03 3D2E58C1 B3A46B6C EC8CE8BB 1415D368 D8F47C6E

6. Verify that $P \neq O$, OK.

7. Set $z = x_P$.

z = 03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503

8. Convert z to an octet string.

$$Z = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503}$$

9. Use the key derivation function ANSI-X9.63-KDF with SHA-1 to generate keying data K of length 20 octets from Z.

   9.1. Append $Counter_1 = \texttt{00000001}$ to the right of Z.

   $$Z_1 = \texttt{03 8359FFD3 0C0D5FC1 E6154F48 3B73D43E 5CF2B503 00000001}$$

   9.2. Compute $Hash_1 = SHA\text{-}1(Z_1)$.

   $$Hash_1 = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

   9.3. Get $K = Hash_1$.

   $$K = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

Output: The keying data K.

$$K = \texttt{49111524 921C9033 3A317C3D 04A5FCD3 D45B2880}$$

6 References

The following references are cited in this document:

[1] SEC 1. *Elliptic Curve Cryptography*. Standards for Efficient Cryptography Group, September, 1999. Working Draft. Available from: `http://www.secg.org/`

[2] GEC 1. *Recommended Elliptic Curve Domain Parameters*. Standards for Efficient Cryptography Group, September, 1999. Working Draft. Available from: `http://www.secg.org/`