A continuación le descripción del proceso que utilizo para obtener el valor de llaveCertificado:

Paso 1: Generar archivos PEM a partir de key y cer:
Con el siguiente código en php (se ejecutó sólo una vez al instalar el sello del cliente en nuestro sistema)

```
<?php
  system('openssl x509 -in 00001000000201532020.cer -inform DER -out
00001000000201532020.cer.pem -outform PEM');
  system('openssl pkcs8 -in mars7001114w6_1207111111s.key -inform DER  -passin
pass:xxxxxx -out mars7001114w6_1207111111s.key.pem -outform PEM');
?>
```

Esto generó los archivos 00001000000201532020.cer.pem y mars7001114w6_1207111111s.key.pem que son los que se están usando actualmente y de manera exitosa para el timbrado

Paso 2: Obtener los datos internos de la llave privada:
Con el siguiente código en php:

```
<?php
  $Comando = "openssl rsa -inform PEM -outform PEM -in
mars7001114w6_1207111111s.key.pem -out 1.txt -text";
?>
```

El comando anterior genera un archivo de texto plano (1.txt) con la siguiente información:

```
Private-Key: (1024 bit)
modulus:
    00:9c:d3:00:13:f9:f7:7f:58:40:16:20:63:13:e2:
    a7:91:75:42:ca:17:1a:0a:e7:1f:81:5c:6f:c2:49:
    f2:ed:60:3c:7c:57:8b:c6:c7:37:58:d1:bd:40:40:
    e6:a8:9c:c0:7c:4d:6a:23:f7:ae:ab:d3:e6:ce:6a:
    05:b0:c7:98:14:6d:ef:9b:e5:c2:d8:17:1f:13:f0:
    d9:5d:ad:42:a3:d4:4e:b6:4d
publicExponent: 65537 (0x10001)
privateExponent:
    7a:d0:99:26:27:d0:47:8e:59:41:fa:c3:1c:c3:97:
    23:36:41:33:1e:ee:65:6c:67:03:ce:5e:85:bc:2b:
    d3:1f:be:57:42:4b:ee:21:6a:35:2b:00:2c:a5:3f:
    28:0e:9b:8a:e2:d7:e7:ff:7f:cc:04:3b:b2:33:ca:
    e7:03:36:45:a9:fe:95:01
prime1:
    00:ce:75:67:31:cf:96:0b:c5:0b:9f:45:67:f1:4b:
```

21:d7:88:e1:b0:76:6f:6c:5c:e5:66:8c:00:21:ed:
01:da:2b:76:18:a6:7d:12:12:98:56:a0:22:c9:7d:
73:8f:f6:74:51
prime2:
00:c2:75:06:18:aa:6b:26:7c:1a:fa:d0:03:36:4c:
31:ca:3a:0f:3c:0a:e6:f3:54:20:d6:2a:77:1a:e5:
71:5b:11:ce:2d:28:bf:19:ed:58:96:fc:62:24:e2:
da:31:b0:4f:3d
exponent1:
7b:64:7d:bd:ae:84:ce:1a:01:9d:3a:7d:2a:20:ae:
64:44:16:27:16:51:f7:e0:f9:96:35:7c:6a:ca:8a:
15:26:be:99:73:00:2c:aa:62:73:fb:97:6e:f7:54:
97:29:44:51
exponent2:
0e:31:dc:b7:12:39:a0:25:8f:12:9f:fc:9c:0a:14:
ce:a6:d6:90:09:33:36:ed:4c:5c:e5:7f:f7:09:ed:
d8:85:44:77:6d:94:4e:4e:e5:d6:bc:62:d5:63:ca:
dd:87:b1:41
coefficient:
55:43:8b:f1:e1:c0:9e:59:65:06:ce:0f:6d:ec:53:
74:fa:88:ad:83:c3:00:0a:f4:f2:8c:6a:09:84:61:
27:3c:7f:63
-----BEGIN RSA PRIVATE KEY-----
MIICWwIBAAKBgQCc01kT+fd/WEAWIGMT4qeRdULKFxoK5x+BXG/CSfLtYDx8V4vG
xzdY0b1AQPQnmRoHSJB30X3yFbJQZmRlQ6TWRSrn/PtZ/QXSyR/5q+SZGGMkps80
UNolceaonMB8TWoj966r0+bOagWwx5gUbe+b5cLYFx8T8NldrUKj1E62TQIDAQAB
AoGAetCZJifQR45ZQfrDHMOX7zYhxBorbvZpmelGLKW09Wk0dTwqXtWM5qX6y+MS
UiB4XdPaL7uEIp4TMxu87AD5iQpQpHsQiPTi6OU+IzZBMx7uZWxnA85ehbwr0x++
UiB4XdPaL7uEIp4TMxu87AD5iQpQpHsQiPTi6OU+IzZBMx7uZWxnA85ehbwr0x++9+D5lj
V8asqKTOMr7q7Qh4/Q6mG7smc/FSa+mXMALKpic/uXbvdUlylEUQJADjHc
txI5oCWPEp/8nAoUzqbWkAkzNu1MXOV/9wnt2IVEd22UTk7l1rxi1WPKUouKMWG4
EXxx6xOhVm/43Yex KA6biuLX5/9/zAQ7sjPK5wM2Ran
lBs4PbexTdPqIrYPDJAr08oxqCYRhg6Os
JNIUov9QKBl1czFpR9yiBguZBmmKtxvFnZrXJzx/Yw==
-----END RSA PRIVATE KEY-----

Paso 3: Integrar el resultado anterior a la programación de la solicitud de cancelación:

Abrír el archivo de texto y copie la información de su código (esto es temporal ya que una vez que funcione tiene que leer la información directamente del comando que abre el archivo key.pem

$Modulus =
"00:9c:d3:59:13:f9:f7:7f:58:40:16:20:63:13:e2:a7:91:75:42:ca:17:1a:0a:e7:1f:81:5c:6f:c2:49:f2:
3:a4:d6:45:2a:e7:fc:fb:59:fd:05:d2:c9:1f:f9:ab:e4:99:18:63:24:a6:cf:34:50:da:25:71:e6:a8:9c:c0:

```
7c:4d:6a:23:f7:ae:ab:d3:e6:ce:6a:05:b0:14:6d:ef:9b:e5:c2:d8:17:1f:13:f0:d9:5d:ad:42:a3:d4:4e:b
6:4d";
   $PrivateExponent =
"7a:d0:99:26:27:d0:47:8e:59:41:fa:c3:1c:c3:97:ef:36:21:c4:1a:2b:6e:f6:69:99:e9:46:2c:a5:b4:f5:
69:34:75:3c:2a:5e:d5:8c:e6:a5:fa:cb:e3:12:52:20:78:5d:d3:da:2f:bb:84:22:9e:13:33:1b:bc:ec:00::
42:4b:ee:27:6a:35:2b:00:2c:a5:3f:28:0e:9b:8a:e2:d7:e7:ff:7f:cc:04:3b:b2:33:ca:e7:03:36:45:a9:fe
:95:01";
   $Prime1 =
"00:ce:75:67:31:cf:96:0b:c5:0b:9f:45:67:f1:4b:21:d7:88:e1:b0:76:6f:6c:5c:e5:66:8c:00:21:ed:4d:
:08:cb:32:df:49:14:61:01:da:2b:76:18:a6:7d:72:12:98:56:a0:22:c9:7d:73:8f:f6:74:51";
   $Prime2 =
"00:c2:75:06:78:aa:6b:26:7c:1a:00:d0:03:36:4c:3:f8:6b:26:7c:1a:0095:d6:ee:32:34:53:bd:08:3c:
3:54:20:d6:2a:77:1a:e5:71:5b:11:ce:2d:28:bf:19:ed:58:96:fc:62:24:e2:da:31:b0:4f:3d";
   $Exponent1 =
"7b:64:7d:bd:ae:84:ce:1a:01:9d:3a:002a:20:ae:64:44:42:27:16:51:f7:e0:f9:96:35:7c:6a:ca:8a:4c::
ce:1a:01:26:be:99:73:00:2c:aa:62:73:fb:97:6e:f7:54:97:29:44:51";
   $Exponent2 =
":12:9f:fc:9a:0a:14:ce:a6:d6:90:09:33:36:ed:4c:5c:e5:7f:f7:09:ed:d8:85:44:77:6d:94:4e:4e:e5:d6:
bc:62:d5:63:ca:52:8b:8a:31:61:b8:11:7c:71:eb:13:a1:56:6f:f8:dd:87:b1:41";
   $Coefficient =
"55:43:8b:f1:e1:c0:9e:59:65:06:ce:0f:6d:ec:53:9a:fa:88:ad:83:c3:24:0a:f4:f2:8c:6a:09:84:61:83:a
3:ac: 47:dc:a2:06:0b:99:06:69:8a:b7:1b:c5:9d:9a:d7:27:3c:7f:12";
```

A continuación viene la parte que anteriormente me estaba haciendo falta:

```
   $Modulus = SSLDataExtractedToXMLData($Modulus, true);
   $PublicExponent = "AQAB";
   $PrivateExponent = SSLDataExtractedToXMLData($PrivateExponent, false);
   $Prime1 = SSLDataExtractedToXMLData($Prime1, true);
   $Prime2 = SSLDataExtractedToXMLData($Prime2, true);
   $Exponent1 = SSLDataExtractedToXMLData($Exponent1, true);
   $Exponent2 = SSLDataExtractedToXMLData($Exponent2, false);
   $Coefficient = SSLDataExtractedToXMLData($Coefficient, false);
   $KeyInXML =
"<RSAKeyValue><Modulus>".$Modulus."</Modulus><Exponent>".$PublicExponent."</Expo
nent><P>".$Prime1."</P><Q>".$Prime2."</Q><DP>".$Exponent1."</DP><DQ>".$Exponent2.
"</DQ><InverseQ>".$Coefficient."</InverseQ><D>".$PrivateExponent."</D></RSAKeyValue
>";
```

La function SSLDataExtractedToXMLData es así:

```
function SSLDataExtractedToXMLData($SSLData, $JumpFirst = false)
   {
   $ArrVals = explode(":", $SSLData);
   $a = "";
   if ($JumpFirst)
```

```
  {
  $inicio = 1;
  }
else
  {
  $inicio = 0;
  }
for($i = $inicio; $i < count($ArrVals); $i = $i + 1)
  {
  $a = $a.hexToStr($ArrVals[$i]);
  }
return base64_encode($a);
}
```

Para los datos Modulus, Prime1, Prime2 y Exponent1 hago que el primer elemento del ArrVals no sea tomado en cuenta a propósito, ya que cuando logré que el resultado de procesar el certificado de prueba con mi código fuera idéntico al que usted me proporcionó fui verificando el comportamiento dato por dato hasta que me dí cuenta que Exponent2 y Coefficient no incluyen el primer dato antes de ":" en el resultado deseado

Según lo que estuve investigando PublicExponent siempre es AQAB por lo que lo asigno directamente.

Finalmente el valor de "llaveCertificado" es lo que está contenido en la variable $KeyInXML