Encryption and Decryption test

- Encryption process on Android

```
        ConfigManager.init("jar://jdigidoc.cfg");
       Log.d("despues de:", "ConfigManager.init");

           /** signed doc object if used */
           SignedDoc m_sdoc;
           m_sdoc = null;

           /** encrypted data object if used */
           EncryptedData m_cdoc;
           m_cdoc = null;

           String inFile = null, outFile = null;
           String certFile = null;
           String recipient = null;
           String keyName = null;
           String carriedKeyName = null;
           String sId = null;

           inFile = str;


           outFile = "cifrado.cdoc";

           certFile = "/mnt/sdcard/chespirito.crt";


           // agregar el destinatario
           try {
                   if (m_cdoc == null){
                           Log.d("m_cdoc == null", "-");
                           m_cdoc = new EncryptedData(null, null, null, EncryptedData.DENC_XMLNS_XMLENC, EncryptedData.DEN
                   }
                   Log.d("Adding recipient", certFile);
                   X509Certificate recvCert = SignedDoc.readCertificate(new File(certFile));
                   if (recvCert != null && recipient == null)
                           recipient = SignedDoc.getCommonName(recvCert.getSubjectDN().getName());
                   Log.d("Recipient", recipient);
                   if (sId == null){
                           int n = m_cdoc.getNumKeys() + 1;
                           sId = "ID" + n;

                   }

                   EncryptedKey ekey = new EncryptedKey(sId, recipient, EncryptedData.DENC_ENC_METHOD_RSA1_5, keyName, car
                   m_cdoc.addEncryptedKey(ekey);


           }catch(Exception e){
                   Log.d("Error adding EncryptedKey: ", e.getMessage());
                   Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
           }

           // ejecutar el cifrado
           try {
                   Log.d("Encrypting file:", inFile + " to: " + outFile);
                   File fIn = new File(inFile);
                   // create a ddoc intermediate file
```

```
                m_sdoc = new SignedDoc(SignedDoc.FORMAT_DIGIDOC_XML, SignedDoc.VERSION_1_3);

                Log.d("Encrypting file:", "paso new SignedDoc");

                DataFile df = m_sdoc.addDataFile(new File(inFile), SignedDoc.xmlns_digidoc13, DataFile.CONTENT_EMBEDDED

                Log.d("Encrypting file:", "paso addDataFile");


                byte[] data = SignedDoc.readFile(new File(inFile));

                Log.d("Encrypting file:", "paso readFile");

                df.setBase64Body(data);

                Log.d("Encrypting file:", "paso setBase64Body");

                byte[] inData = m_sdoc.toXML().getBytes("UTF-8");

                Log.d("Encrypting file:", "paso toXML()");

                Log.d("Encrypting file", "Content: " + inFile + " size: " + data.length);
                Log.d("Encrypting file", "DF: " + new String(inData));

                m_cdoc.setData(inData);
                m_cdoc.setDataStatus(EncryptedData.DENC_DATA_STATUS_UNENCRYPTED_AND_NOT_COMPRESSED);
                m_cdoc.addProperty(EncryptedData.ENCPROP_FILENAME, inFile + ".ddoc");
                m_cdoc.setMimeType(EncryptedData.DENC_ENCDATA_TYPE_DDOC);
                StringBuffer sb = new StringBuffer();
                sb.append(fIn.getName());
                sb.append("|");
                sb.append(new Long(fIn.length()).toString() + " B|");
                sb.append("application/unknown|");
                sb.append("/" + fIn.getName());
                m_cdoc.addProperty(EncryptedData.ENCPROP_ORIG_FILE, sb.toString());
                //m_cdoc.addProperty(EncryptedData.ENCPROP_ORIG_SIZE, new Long(inData.length).toString());

                int nCompressOption = 0;

                m_cdoc.encrypt(nCompressOption);


                // genera el archivo cifrado en /data/data/ve.gob.cenditel/files
                FileOutputStream fos = openFileOutput(outFile, Context.MODE_PRIVATE);

                Log.d("Encrypting file", "antes de escribir archivo " + outFile);

                fos.write(m_cdoc.toXML());

                Log.d("Encrypting file", "despues de escribir archivo " + outFile);

                fos.close();

                Log.d("Encrypting file", "despues de cerrar archivo " + outFile);

                Toast.makeText(getApplicationContext(), "Cifrado correctamente: " + outFile, Toast.LENGTH_SHORT).show()


        } catch(Exception e) {
                Log.d("Error encrypting file: ", inFile + " - " + e.getMessage());
                e.printStackTrace(System.err);
```

```
                    Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH_SHORT).show();
            }
```

- Decryption process with jdigidocutil (encrypted file through Android)

```
aaraujo@moe:~/desarrollo/andriod/pruebas/jdigidoc-3.7/jdigidoc$ java -jar jdigidocutil-\$\{version\}.jar -cdoc-in /tmp/cif
JDigiDoc - 3.6.0.1
Reading config file: jar://jdigidoc.cfg
Reading encrypted file: /tmp/cifrado.cdoc
Decrypting to: /tmp/salida.pdf
Using recipient: 0
SAXDigiDocFactory::readSignedDocOfType
Start reading ddoc/bdoc from file: /tmp/salida.pdf
Start Element: SignedDoc lname:  uri:
Start Element: DataFile lname:  uri:
Start collecting digest
Allocating buf: 122556 Element: DataFile lname:  uri:
Attr: ContentType ='EMBEDDED_BASE64'
Attr: Filename ='LSMDFE.pdf'
Attr: Id ='D0'
Attr: MimeType ='http://www.sk.ee/DigiDoc/v1.3.0#'
Attr: Size ='61278'
Attr: xmlns ='http://www.sk.ee/DigiDoc/v1.3.0#'
Canonicalized: '<DataFile xmlns="http://www.sk.ee/DigiDoc/v1.3.0#" ContentType="EMBEDDED_BASE64" Filename="LSMDFE.pdf" Id=
End Element: DataFile collect: 1
Digest: D0 - 5hmr2x1Q4PKrjbmkWy7K+jzYL/o= size: 61278
End Element: SignedDoc collect: 0
JDigiDoc end, time: 3 sec result: success
```

An example of a file without xmlns prefix

```
<?xml version="1.0" encoding="UTF-8" ?><EncryptedData MimeType="http://www.isi.edu/in-noes/iana/assignments/media-types/ap
DzANBgNVBAgTBk1lcmlkYTEPMA0GA1UEBxMGTWVyaWRhMREwDwYDVQQKEwhDRU5E
SVRFTDEOMAwGA1UECxMFR0lEU0kxLjAsBgNVBAMMJUF1dG9yaWRhZCBkZSBDZXJ0
aWZpY2FjacOzbiBkZWwgR0lEU0kxJjAkBgkqhkiG9w0BCQEWF2FjZ2lkc2lAY2Vu
ZGl0ZWwuZ29iLnZlMB4XDTEyMDQyNjE2MDUwMFoXDTEzMDQyNjE2MDUwMFowgZYx
CzAJBgNVBAYTAk1YMRIwEAYDVQQIEwlNb250ZXJyZXkxEDAOBgNVBAcTB0phbGlz
Y28xETAPBgNVBAoTCENFTkRJVEVMMQ4wDAYDVQQLEwVHSURTSTETMBEGA1UEAxMK
Q2hlc3Bpcml0bzEpMCcGCSqGSIb3DQEJARYaY2hlc3Bpcml0b0BjZW5kaXRlbC5n
b2IudmUwggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDNWsKCFNPW4Cxq
qduuzbMODqUbt7uSAxdnyLTpZ6P8cfwnberSzq/8X2kKTpuzScb2L4+0I4GxAaqu
td80XJInKBtcTqYpzMpb3HlPbEUr1l+BbkUQ4Jn4CnE60UNY9d4ebTEOyV4LcuRZ
/fjnapeFppWYrJ5kOjtRCGlbSl8azP4NCMhh6alrhW2JMwV6AmvQTgk2m8tzIDzO
fM9YcHuqU31ly2vzLbj+xc+oSevUftwdoEtU6bogCrXuDnBsEmPNqUacDk+o1zj8
vmWXYByTudsfKcG1zqwHtSOcK3neb9XU8gNCLPsZpNnG2tGT3FfUZTn7zi6YRcD2
OFJZ4rClAgMBAAGjbzBtMAwGA1UdEwEB/wQCMAAwCwYDVR0PBAQDAgTwMBEGCWCG
SAGG+EIBAQQEAwIFoDAeBglghkgBhvhCAQ0EERYPeGNhIGNlcnRpZmljYXRlMB0G
A1UdDgQWBBRhVVfxfHWbQhFkC0qK0wkyonrTaTANBgkqhkiG9w0BAQsFAAOCAgEA
Xq9j6jVCt7zdwv0AAUpSdQtMR7bMnUvy3eu+C7omW/SZxrg4EYYgn0iI+rgj9A+O
0R0QcG4TgM5dnUtWae4PKqQEUH8tbc8jvERlD0gxXGDWU3KEXJ0UBDgmwr57eI9B
sOSG4Y4rKB/PEsrLUpODfY+rxDDPoU+R+8FdBzUl3QLsgAWPHkbWDLM7xpmMJiMx
umTckoGn8riv1FvJ16t4UJYqIsETdRpoIr51yQ0m+mzfKji/cgpo6HFG2f0BmC+G
G7ma1PWl2Liaf9Y8J+kmqp9aX1JHcAX7H7tpgC9KUxcmdWrzORaLjNTz2KjgcoWj
o/TWoQDH6vNXCZq5bInmvn4BARNAxaZQMEkp3K6uKlgHJwKfHwsxf68xnC9InMV+
mfQWA1Ls3UbHm6gZjDxQL4a8y6NCB66RbYBFJRWh56aY/FxEbZ4gNwkFiEZO3Yxg
hv21gJyBvh89l2uExyOZn4WJUrTBYlb348o4G9ZvXA3kfklG3B+rz+24qLS5iWm4
bsEkQg0nnuQmB1CTJTSavjbmzZ4gDavCdgXVHmeeLYoNLb1YUMSDt1pENCJ8BZR0
n4dQDUQiP8dIqmpoGK9R1wasxijb99QnQXDr++55nATijn1qcssqYV2c4R1b5Gri
```

0GlIiOlnCN9r1us1Icr+9vU4Z3qao4okTSQrVjSm/og=</X509Certificate></X509Data></KeyInfo><CipherData><CipherValue>lJmQmmDM4t4/OE
2lr7kCKxOmoMuiJqTPEwLg/5KFgm+1oBdP131sM8dYH3DC/3s82O5tyijpg1K7wW
i0Mp9dduIqlq+x1pdMZlTA76I3q+/gi29e8FNgDaSYToNTuD2xAXjeUDQrkDTjMR
kk6gzwQkQgwPDIbblSZtIuL+JbLDsoP9sYiSrQXUh1M2cSsOp7bfURCmXHDUv1GD
3ZtvwkFvJRJuFKkbEmn/xn4qHvP/GGDTJWLM145IX8O1eGLY1vJAe/vnYXx3UWHW
us62TufDpDgv3I0Knx8xOA==</CipherValue></CipherData></EncryptedKey></KeyInfo><CipherData><CipherValue>Mg++HduoRKVYMHCOqF9U0
9R/Mp7eoBkNi0eBoISIzTigpgHrMwJcEzcXhQPxfNsLrrTsuTHO60douVNbXzxvg
hR4dv1Dd3QZDAe6xK/COEQKukiRbVlys3i6B9427yFnAHs5Y3d8vLMbHdpeu3W8x
Pw/N3QPWgtoSujsdQ8y6u/ZFnyxFaUbIgNFdC7vhY/xAhSRE7mTHalvm8s45xjIZ
8eX7pyxK56KjtBIjZWw8ZPjFBZEOh33tfefXBycUOz6MUni/9zdgmF2HjfYiATkx
affu3uaKYBKjz6QXcB/rJFKbJPMa8TyDZ+Sm4eq9gfDOkrJy3rLjm14s4722BUfM
zecLg01zDa40p5iTutQZsRODbRznwQ+/swe27xhUrbZj5Gk86w+OP/hLN2YEsrjw
BkFsJD4Z1+aBFrvlk9bSrHIFvBqB+PnrxENObF3M2m8p9anZzNdfXA/mRobHfvG3
6J+tTDMA4sBJyhOd+NgRBcZdSQB/CF8dmma2gZL+vGIsPwfKiambiRs2Vzm0tKMx
c1F/13tUQ71qmjVgKGDR9yx59JKD3OxYw3N3TQFUmoCc59U+4djWFUt/A3OQTC48
pSGfKNodGEChFtR/ks8Vbq1vqAP0fs9WEXs1XctqZjIaR9TG7NJoP4Y/i+Ldu+jB
zuikaV0uXmrNOtK27Z0IZS1reOPoe9496VxMHFX+fTrUF7yb877AyCSPAENHVCrY
inU4cItIyKmbly0001VaAmOgiTI1jfF4nIJp6yG5CJydet0/p5kPkXi7hpX4iBzH
uuUnBvvZdcUz7oSssiea/GBNxus2B+EdFI2LeE8Tg67SKSQUD/CpgdOdj6x8n7gv
FgULByAM3Rv2NUfNBIug9dQPY99tqYt3FYTgonethmB2h89q1YnYbKoeLXnrIQRx
dSq8CXBcdREHC3zsmqXv4cjkrqVQiz6CeGPMyzJbeXJDo0ZtiZWBf1x7TOSI8FfY
wWS3FMgIoEUCh0+ZtwI9fK9HJn5BqtCRVHXc5OmJli81sPmMUkgvwGvaSNodPv40
/5Tour5aPwXMaMJB8XTKmxK7xrxkMGCGiC6R0q9B6R4VDsj/nevqKpRAQGEF7+G3
SyLrLRSruZ30WhszIgBrsSgiH8VoiZHLKUOdHgW0kAO7S+3q2UlDAxAidgCgTD5S
eqeYjBImf2dTQ+B6fokeqfzZ1sbFEdjDnv+YjAY5i2aNIhqzlbE4cX7JgNu497Bn
lzOI27UB51ExbyJ68Lvm0K3yKpyMn8KgS0haYkWIGAl2Q7u1wKd2skyW6eKbQ66y
aZu+oouotxMAMLco6dL/gjezTpiCWUILomYi4Ud6Hmk6ExmqAjccNziQueBEHT07
OMet5nWv2D4KVVh9+2+gZw==</CipherValue></CipherData><EncryptionProperties><EncryptionProperty Name="LibraryVersion">JDigiDo