Cryptlib cryptographic library

What it is?

Cryptlib is a powerful cryptographic library which provides implementation of complete secure services such as S/MIME, PGP/OpenGPG secure enveloping, SSL/TLS and SSH secure sessions, CA services such as CMP, SCEP, RTCS and OCSP and other security operations such as secure timestamping (TSP). Cryptlib uses many industry standards, so it's not tied to a single platform/system/library. It enables you to use many crypto devices like hardware accelerators, HSMs, cryptosmart cards and others. This library has binding to many languages like C/C++, C#, Java or Python. Source codes of the library can be compiled under many operating systems, e.g. UNIX, DOS, Windows, Xilinx XMK, uClinux. It provides interface to many popular encryption algorithms, but provides high level interface, so it hides you from most of the implementation details, although low level encryption routines can be used too. It takes care of encoding issues and cross-platform compatibility problems.

Addresses, Version, License

Crpytlib can be downloaded in the form of source codes from http://www.cs.auckland.ac.nz/~pgut001/cryptlib/ (or from http://www.cryptlib.com). The library is distributed under dual license, that allows free, open source use under a GPL-compatible license and closed source use under a standard commercial license (for large scale commercial use). The current version of the library is 3.3.2.

Using the library

Include "cryptlib.h" in your applications and link with –lcl under Unix and add cl32.lib to your project in Windows.

Typical usage

S/MIME – cryptlib employs the IETF standardized Cryptographic Message syntax (CMS, formerly called PKCS #7) format as its native format. The S/MIME implementation uses crpytlib's enveloping interface allow simple integration of strong authentication and encryption capabilities into existing messaging/email software. You can simply push data into envelopes using cryptlib's interface and on the other side pop the data back out using only three function calls (plus one to add signature or encryption key).

PGP/OpenPGP – this messaging format allowes you to send or receive PGP encrypted email and data. It can be used as simple as in case of S/MIME.

Secure sessions – cryptlib provides both client and server implementations of all session types (TLS, SSL, SSH).

Cryptlib can take care of key management too:

Certificate management – cryptlib implements full X.509 certificate support, including all X.509 version 3 and X.509 version 4 extensions. It supports additional certificate types like SET certificates, Microsoft Authenticode and others. It allows generation of

certificate requests suitable for submission to certificate authorities. It's also possible to use cryptlib to provide full CA services. Cryptlib can import and export certification requests, certificates, certificate chains and CRL's, covering the majority of certificate transport formats used by a wide variety of software such as web browsers and servers. It provides also some logging/auditing facilities. The CA keys can be optionally generated and held by tamper resistant hardware security modules. Certificates can be stored in many usually used databases using RDBMS.

User interface – Cryptlib includes a few user interface components as well. They simplify working with keys and certificates.

Sample code:

```
#include "cryptlib.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main() {
      FILE *file;
      char buffer[1000];
      int p;
      int bytesCopied;
      CRYPT_ENVELOPE cryptEnvelope;
      CRYPT_ENVELOPE cryptEnvelope2;

      file = fopen("testfile", "rb");
      if(!file)return 1;
      p=fread(buffer,1,9999,file);
      if(!p) return 1;
      fclose(file);

      cryptInit();
      cryptCreateEnvelope(&cryptEnvelope,CRYPT_UNUSED,CRYPT_FORMAT_CRYPTLIB );
      cryptSetAttributeString(cryptEnvelope,CRYPT_ENVINFO_PASSWORD,"password",
8);
      cryptPushData(cryptEnvelope, buffer, p, &bytesCopied);
      cryptFlushData(cryptEnvelope);
      cryptPopData(cryptEnvelope, buffer, 10000, &bytesCopied);
      cryptDestroyEnvelope( cryptEnvelope );

      file = fopen("envelopedfile", "wb");
      fwrite(buffer,1,bytesCopied,file);
      fclose(file);

      cryptCreateEnvelope( &cryptEnvelope2, CRYPT_UNUSED, CRYPT_FORMAT_AUTO);
      cryptPushData(cryptEnvelope2, buffer, bytesCopied, &bytesCopied);
      cryptSetAttributeString( cryptEnvelope2, CRYPT_ENVINFO_PASSWORD,
"password", 8);
      cryptFlushData(cryptEnvelope2);
      cryptPopData(cryptEnvelope2, buffer, 10000, &bytesCopied);
      cryptDestroyEnvelope(cryptEnvelope2);

      file = fopen("denvelopedfile", "wb");
```

```
        fwrite(buffer,bytesCopied,1,file);
        fclose(file);

        cryptEnd();
}
```

Assignments:

1) Use enveloping (set key, envelope, denvelope, save file) with IDEA algorithm and a fixed key {3}
2) Create a DSA key-pair {3}
3) Digitally sign data with the key generated in step 2 using PGP/GPG enveloping. {4}