The Libgsl Reference Manual

Version 1.0-gitb7c17fd
29 June 2012

Grégory Alvarez (greg@goswell.net)
Charles Berenguer (charles@goswell.net)

Table of contents

Disclaimer :

This is the first public release of the GLS library, it is open source under the GNU Lesser General Public License. The objective was to demonstrate that it worked, optimized it will be the subject of a next release. See the «What's next ?» part for more information.

Libgls depends on the libraries 'libgpg-error', 'libgcrypt' and 'libtasn1'. These libraries are under the GNU Lesser General Public License. The text of the license can be found in the section entitled "GNU Lesser General Public License".

1 - Why GLS ?

The security of an information chain is equal to its weaker link. What I call chain of information is the path traveled by data between a service and its user. Today everybody is trying to secure both extremities of this chain with multiple process like firewall, antivirus, ASLR... But what happen when the links connecting both extremities are neglected ? The information chain security fall down.

SSLStrip

We were working on a client-server project and I wanted to implement a secure connexion between them. I first thought about «TLS» but something wasn't right about it. I knew that TLS had a long history with security problems. Take a look at the BEAST attack made public at the Ekoparty 2011 or at the DigiNotar incident that shows the current problem with certificate. But the security problem that stroke me most was illustrated by a script made by Moxie Marlinspike : SSLStrip. It intercepts HTTPS connections and disable TLS on the victim's side. TLS remains active between the server and the attacker.

I know what you will say, it makes both the S of HTTPS disappear on the navigator bar and the padlock pictogram. But how many people know what they mean and check them ? And the worst part is that lots of websites switch between HTTPS and HTTP for login purpose and you can't see it. SSLStrip works also very well with application like Mail for example, if it can't find TLS it switches by default under a non secure connexion, and again, there is no notification. A funny thing is that a lot of secure payment module have a padlock picture, that doesn't mean anything, and still appears using SSLStrip. This misleads even more the user.

The fact that someone can intercept a secure connexion using TLS and disable it that easily was really intriguing. I did some digging into how TLS works to know how it was possible. When I found the answer I compared my results with all the secure communication protocols to find out that they all work the same way.

The problem comes from what every protocol does: Key Exchange. This permits SSLStrip to get between the victim and the server. If TLS didn't use the Key Exchange protocol, it would have been impossible for SSLStrip to intercepts HTTPS connection.

Yes you can use use a trick, i.e., authenticating your TLS Client with a signed certificate. But it's only a trick and it changes the user's behavior, which is bad and will never be widely implemented. Authenticating your TLS client also complicates everything and lets the security be reflected by the actual certificate system which is not great.

SSLStrip illustrates very well that there is a problem with secure communication protocols. If someone can easily disable your encrypted connexion, how can it be secure ?

Another problem I noticed working on TLS is that every clients send is its password to the server for authentication purpose. It is practically always done in plaintext so when someone intercepts a connexion it is very easy for him to retrieve it. An idea would be to

hash it before sending it, preventing an attacker to gain too much informations. The real and simple solution if you don't want the user's password to be intercepted is to never send it to the server.

MS-CHAP v2

MS-CHAP is a protocol developed by Microsoft to provide authentication derived from the Challenge-Handshake Authentication Protocol. It was revised in 2000 with the version 2. Today this protocol is mostly used in PPTP's VPN and in WPA2 Enterprise.

However this protocol had always had some security issues, as said by Bruce Schneier, Mudge and David Wagner in their 1999 paper :

« In this section we present a very serious attack on the way that exportable 40-bit session keys are generated. This weakness is also present in MS-CHAPv1 as well as MS-CHAPv2, but it has not been discovered until now. The end result is that the so-called "40-bit keys" really only have an effective strength of about 26 bits. As a result, the export-weakened protocol can be cracked in near-realtime with only a single computer. »
Bruce Schneier - Cryptanalysis of Microsoft's PPTP Authentication Extensions

At Defcon 20, David Hulton and Moxie Marlinspike presented a new attack reducing the key space to search to 2^{56}. They also announced a new service to provide the computational power to do an exhaustive attack in less than one day.

I'm not going to explain in details this attack, you can find more informations at the Defcon 20 video here. This just illustrates again the problem with key exchange protocol.

Random-Number Generator

I wondered why does every secure communication protocol use key exchange ? The simple answer is to use a random session key for every connection. But encrypting your hard drive / files with a single key (derivated from your password) with AES 128 / 256 bits is considered secure. Shouldn't be the same using a communication protocol ? Of cause if you use « 1234 » as a password no security on earth will protect you, random session key or not. As said by Kerckhoffs :

« A cryptosystem should be secure even if everything about the system, except the key, is public knowledge. » Kerckhoffs's principle

One way to see the Kerckhoffs's principle is that the password security reflects the security of the system. I recently read an essay from Bruce Schneier posted on Hackers News about random-number generator and what he said blew me away. The article is 5 years old and no one is able to prove if it is true or not but this is a good conspiracy hypothesis and shows some weaknesses of the key exchange protocol :

« Random numbers are critical for cryptography: for encryption keys, random authentication challenges, initialization vectors, nonces, key-agreement schemes, generating prime numbers and so on. Break the random-number generator, and most of the time you break the entire security system. [...] Generating random numbers isn't easy, and researchers have discovered lots of problems and attacks over the years. A recent paper found a flaw in the Windows 2000

random-number generator. Another paper found flaws in the Linux random-number generator. Back in 1996, an early version of SSL was broken because of flaws in its random-number generator. » Bruce Schneier - http://www.schneier.com/essay-198.html

Since a RNG generates the session key, the security of the communication protocol depends on the actual randomness of the random-number generator. You break or put backdoor in the RNG and you break the security of the communication protocol.

« Which is why you should worry about a new random-number standard that includes an algorithm that is slow, badly designed and just might contain a backdoor for the National Security Agency. » Bruce Schneier - http://www.schneier.com/essay-198.html

So, why would you use a key exchange protocol ?

Knowing that no secure communication protocol suited my needs, and more importantly why, I decided to develop one for the company who will not depend on Key Exchange Protocol.

How it works

Description

GLS, acronym for Goswell Layer Security, is a secure communication protocol developed by the Goswell company to respond to the actual secure connexion protocol's problems.

These problems originate from the key exchange protocol who negotiate the key encryption for the session between client and server. This protocol allow different active attack on the secure connexion.

GLS solve this problem using a know information by both party of the connexion, the user's password. It derivate a key from it to use as encryption key, this way :

- The user's password never travels on the network and still provide an authentication for the server. If the password is wrong, it will not be able to decrypt the message sent by the user and the login will fail.

- There is no key negotiation, both part already know it making active attacks like SSLStrip useless.

The original use of the key exchange protocol was to have a different key encryption for every session to make it harder for an attacker to decrypt the communication. Since no password and encryption key travel under the network, using one only key don't create a security risk. In this condition only an exhaustive key search attack, alias brute-force, is theoretically possible. A 256 bits AES encryption will take hundreds of years to brute-force.

« AES permits the use of 256-bit keys. Breaking a symmetric 256-bit key by brute force requires 2^128 times more computational power than a 128-bit key. A device that could check a billion billion (10^18) AES keys per second (if such a device could ever be made - as of 2012, supercomputers have computing capacities of 20 Peta-FLOPS, see Titan. So 50 supercomputers would be required to process (10^18) operations per second) would in theory require about 3×10^51 years to exhaust the 256-bit key space. »

Source : http://en.wikipedia.org/wiki/Brute-force_attack

To make the server's search for the password easier, a plaintext message is sent from the client to tell is user's ID. All the message following this one will be encrypted.

Registering

To use GLS, the user needs to be registered on the server so both part of the connexion know the same shared secret : the user's password. This condition makes impossible the use of GLS for an unregistered user because he doesn't have a password.

The GLS library supply a function to send a secure message to the sever for registering a user. This function is base on certificate X.509 and public / private RSA keys. This has to be done only once.

Even if this method seems secure, the problematic behind certificate is the trust of a third party who often revealed not to be secure. For this reason it is preferable to handle a user's registration this way to minimize any security risks :

- The user fill 3 informations on the application (email, phone number and a random number who he choose).

- These informations are send using the GLS registration function (asymmetric encryption using the X.509's server certificate, see further for more details).

- The server decrypt the message and send 2 temporary passwords to the user, one by email, one by SMS.

- Once the user receives the 2 passwords, he connect to the server using the 3 informations he have (temporary password from the email, temporary password from the SMS and the random number). This 3 informations are used by the server and the client to derive a password for GLS :

Password = SHA-512(Password Email) + SHA-512(Password SMS) + SHA-512(Number)

- The user change is password (the GLS library allow to change the password on the fly, no need to reconnect).

This is just an advise to minimize the risk that carry certificate. Making the hypothesis that the third party is not to be trusted, it will be very difficult to intercept the connexions on two different network to access the 3 informations. The GLS library only supply the asymmetric encryption/decryption function (including a certificate validation function). The SMS and email sending functions are to be handle by the server.

The application can also directly send the password via the GLS registering function but this will not make much difference with a key exchange protocol except that the secure connexion is forced by the GLS client's library and this has to be done once.

A good solution is to implement both and let the user decide witch one he want to use.

X.509 Certificate

The X.509 certificates are handled by the library on a «non conventional way». This was a personal choice driven by the actual condition of the certificate's system.

The application's stores introduced a new approach for downloading and updating application. These stores supply a totally new distribution system who permit to solve different certificate's problems.

Roots certificates are normally stored by your operative system on your computer. This means if you want a valid certificate you have to deal with these certificate's proprietary for an annual fee, imposing to trust this third party. Another problem is the cross-platform compatibility of the certificate who depends on the operative system.
Instead of trying to have a valid certificate from a third party or trying to distribute a new root certificate on all the operative system, create a root certificate for the application and hardcode it. This way it will always be cross-platform, no fee, no trusted third party. If you need to change it just update the application. The GLS library provides a function to add a root certificate from a file or from a string (hardcoding it on the app).

Google researcher Adam Langley add a nice idea for handling Certificate Revocation List, saying that these mechanisms (CRL, OCSP...) are broken and the good solution is to maintain a local list of revoked certificate. Update mechanism of the different application's store permit to maintain an hardcoded CRL on the application. For these reason the GLS library doesn't provide an OCSP or CRL protocol but instead a function to hardcode a list of revoked certificates.

The certificate validation methodology also change from the conventional way to be more simple and efficient. There is no more path, only a root certificate signing another certificate. The library does these steps :

- Check if the server's certificate isn't in the CRL

- Check the validity date of the server's certificate

- Check if the root certificate signed the server certificate

- Check if the root certificate is valid (signed by his private key)

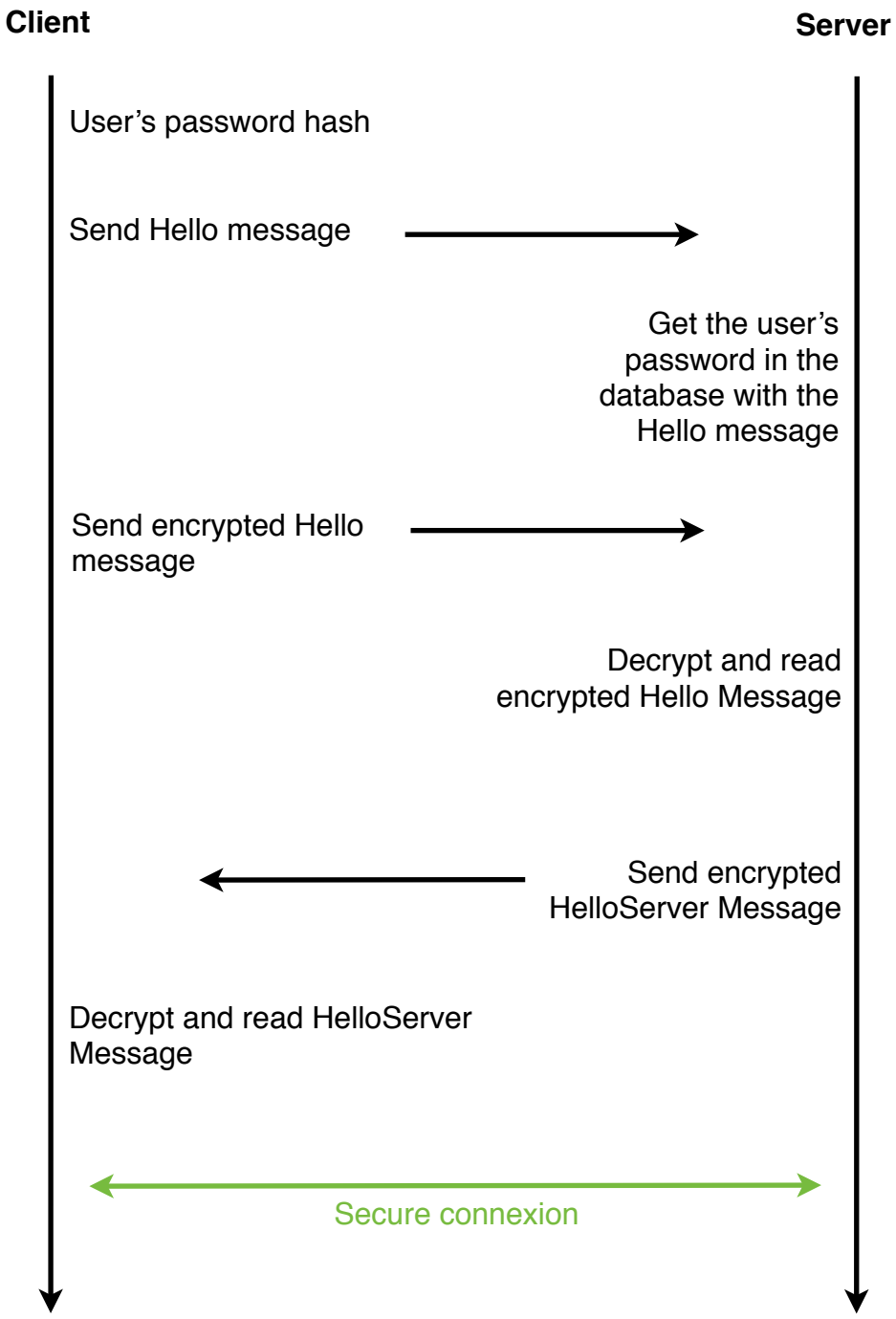
All this choice were made to simplify and secure the certificate's system. However, this also remove any third trusted party obligating to secure the access to the root certificate's application.

In further releases it will be possible to add multiple root certificate to prevent security problems and be able to easily remove one without interfering with the user experience. See the «What's next ?» part for more information.

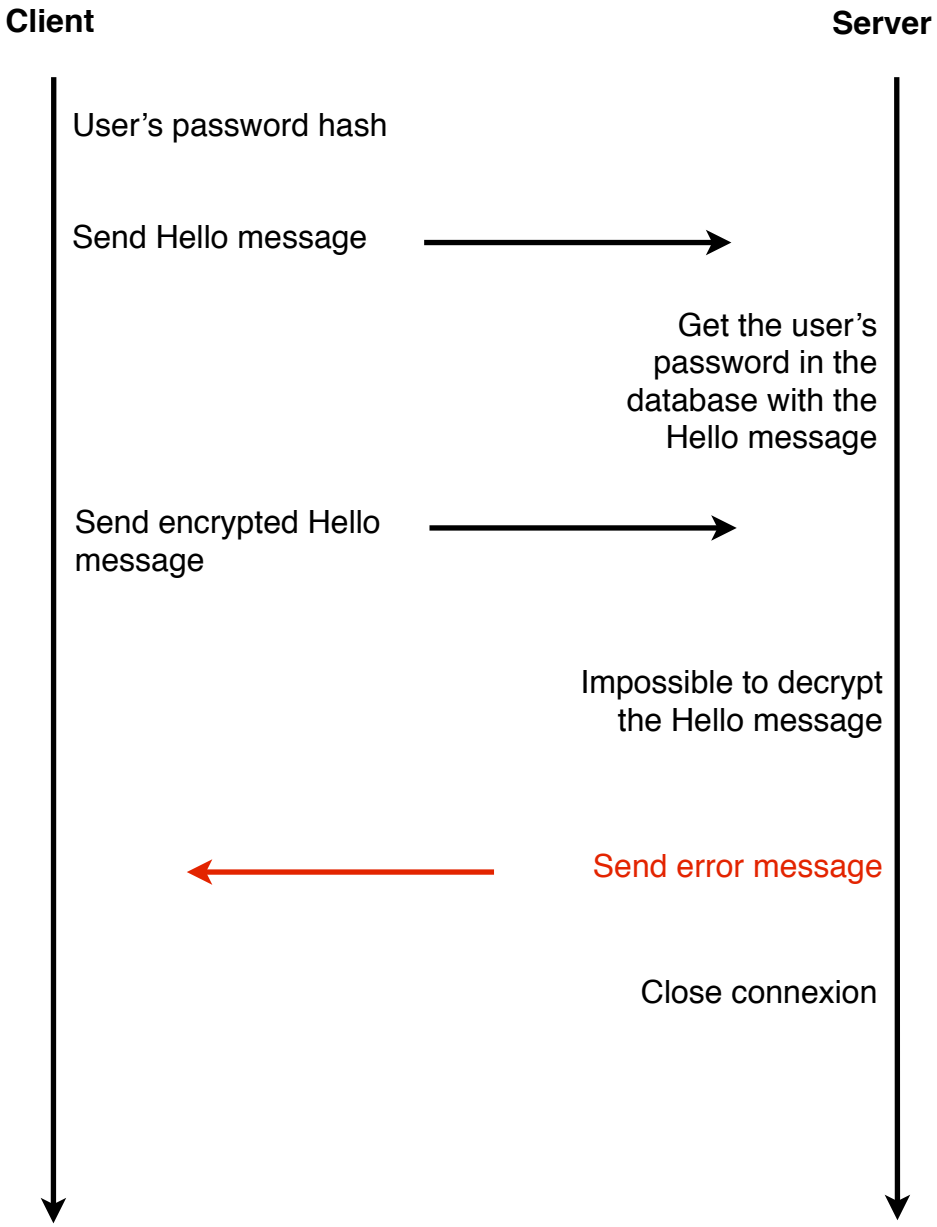
The current library only support certificates with RSA keys and SHA-1 signing.

Negotiation Schema

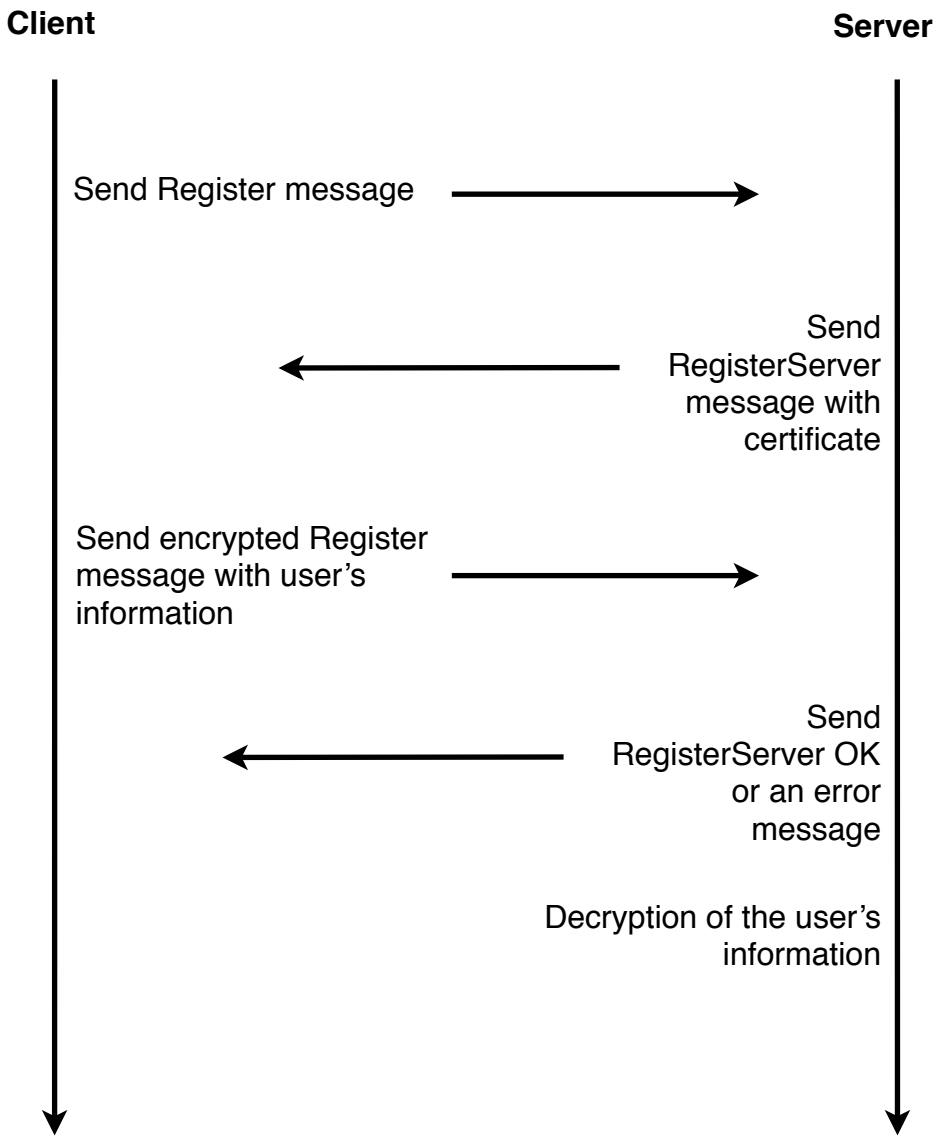
Successful connexion schema :



Failed connexion schema :



Register schema :



The register's informations send by the user are only decrypted by the GLS library and directly transmitted to the application. The library don't handle the registration process.

Technical choice

Today, only a few good and secure encryption algorithm are publicly available : AES (Rijndael), Blowfish, Twofish and Serpent. For a compromise between security and performance, two of them have been chosen to be used in cascade encryption. Twofish is a little bit slower than AES (Rijndael) but faster than the other AES finalist. However, with a 256 bits key it is the fastest. AES (Rijndael) and Serpent are both a substitution-permutation network and Twofish / Blowfish came from Feisteil network.

To vary the algorithm's encryption logic Twofish and Serpent have been chosen. Informations are first encrypted in Serpent (CTS operation mode, 32 rounds, 256 bits key) and then in Twofish (CTS operation mode, 16 rounds, 256 bits key) with both keys derived from the password.

Serpent has a more conservative approach to security than the other AES finalist, the designers knew that 16 rounds was sufficient against actuals attacks but specified 32 rounds to prevent future attacks to be efficient. Some rumors say that Serpent hasn't been chosen for AES because it was too complex for intelligence agencies to break. For example AES (Rijndael) is often used with 14 rounds in TLS whereas Serpent needs to be reduce at least at 9 rounds to supply the same level of security. **Serpent is often considered as the most secure encryption algorithm publicly available.**

In 2003 the NSA announced that a simple AES encryption was sufficient to protect classified information.

« The design and strength of all key lengths of the AES algorithm (i.e., 128, 192 and 256) are sufficient to protect classified information up to the SECRET level. TOP SECRET information will require use of either the 192 or 256 key lengths. The implementation of AES in products intended to protect national security systems and/or information must be reviewed and certified by NSA prior to their acquisition and use. »

Source: http://web.archive.org/web/20070927035010/http://www.cnss.gov/Assets/pdf/cnssp_15_fs.pdf

Encryptions keys are derived using SHA-512 (Secure Hash Algorithm). This function was designed by the National Security Agency (NSA) and publish in 2001 as a Federal Information Processing Standard. It produce a 512 bits hash divided into two parts and used as encryptions keys. The security risks involving the SHA-2 algorithms (SHA-224, SHA-256, SHA-384 and SHA-512) like collision or preimage attack don't influence the GLS protocol's security because no hash is send over the network (an attacker can't intercept them).

Password's hashes

The password is hashed with a SHA-512 function, the result is divided into two 256 bits keys **key1** and **key2** (key1 + key2 = hash). Example :

SHA-512(Password)=91ea1245f20d46ae9a037a989f54f1f790f0a47607eeb8a14d12890cea77a1bbc6c7ed9cf205e67b7f2b8fd4c7dfd3a7a8617e45f3c463d481c7e586c39ac1ed

key1 = 91ea1245f20d46ae9a037a989f54f1f790f0a47607eeb8a14d12890cea77a1bb
key2 = c6c7ed9cf205e67b7f2b8fd4c7dfd3a7a8617e45f3c463d481c7e586c39ac1ed

Using a double authentication it's the hash's concatenation of all the hashes who is divided into the encryption keys.

SHA-512(SHA-512(Password) + SHA-512(File...) + ...) = key1 + key2

Encryption

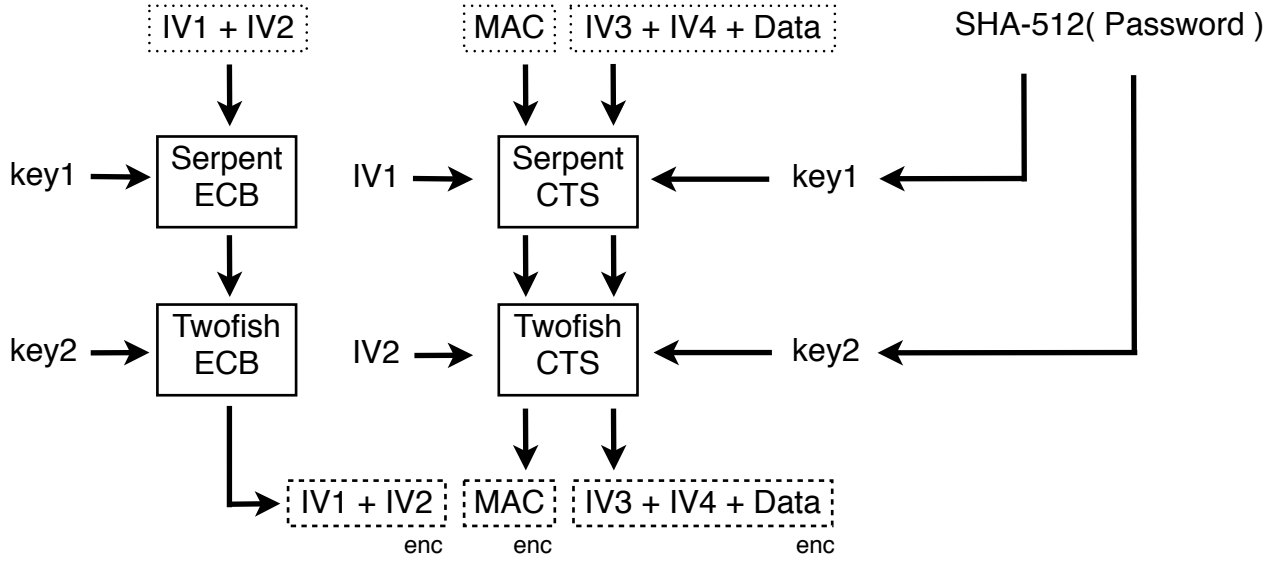
The informations traveling under the network are cascade encrypted by blocks of 128 bits using Serpent (CTS operation mode, 32 rounds, 256 bits key) and then Twofish (CTS operation mode, 16 rounds, 256 bits key).

To prevent replay attacks, Initialization Vectors are synchronized and encrypted. Each IV (IV1, IV2, IV3 and IV4 of 128 bits) is random and unique. When a message arrives the IVs corresponding to it (IV1 and IV2) are compared to the next IVs of the anterior message (IV3 and IV4). If they don't match the message is ignored. The Initialization Vectors of the first message are encrypted in ECB. This way every party of the connexion always know the IVs for the actual message and the IVs of the next message.

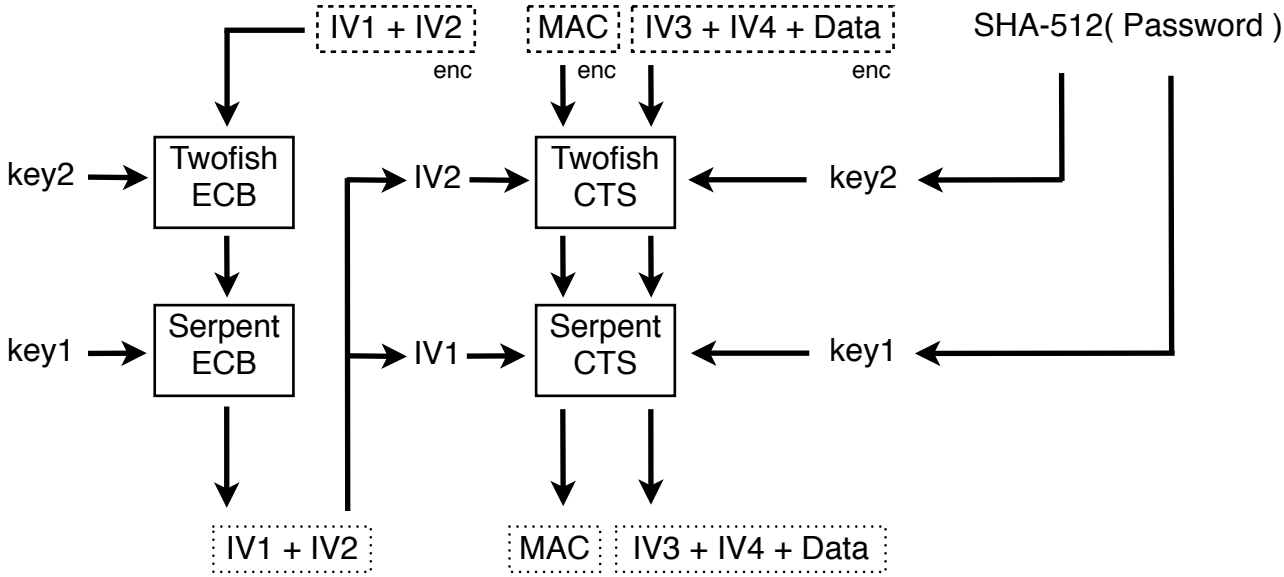
A Message Authentication Code (MAC) in SHA-256 is also encrypted and send with the message.

Encryption schema

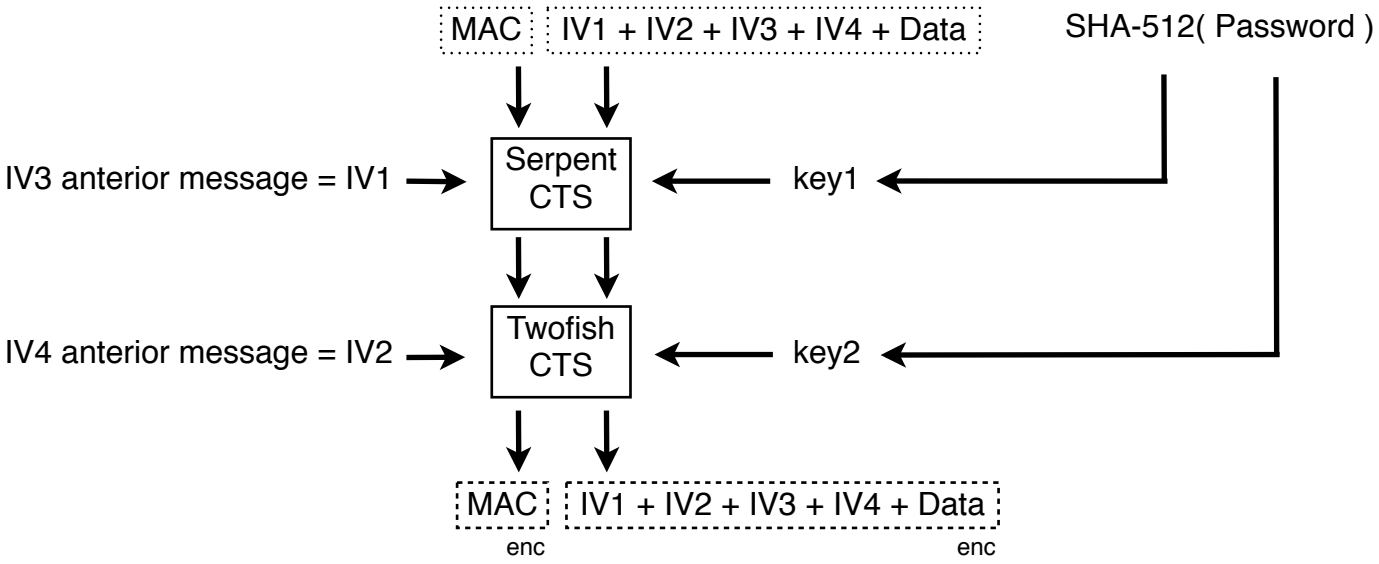
First message encryption :



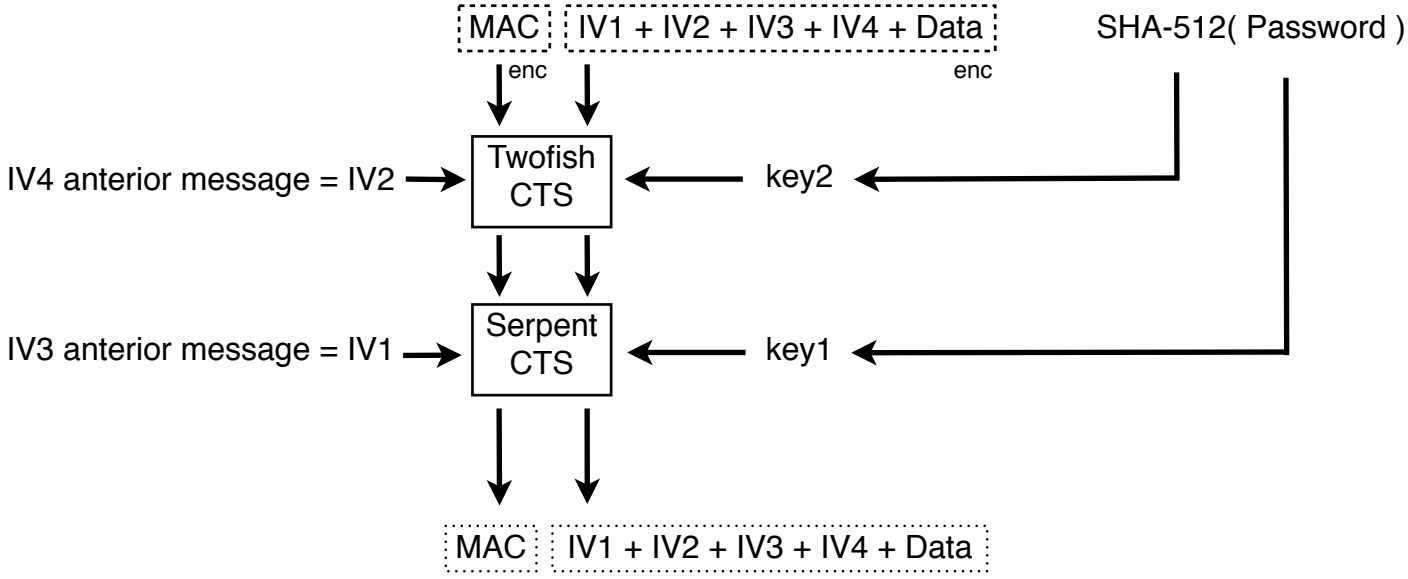
First message decryption :



Encryption of all others messages :



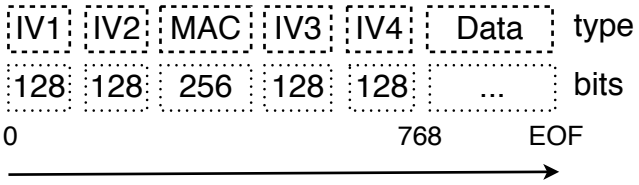
Decryption of all others messages :



The Initializations Vectors IV3 and IV4 are always taken from the last message without consideration of it's state (if it was send or received). For example when sending multiple following message it's the IVs from the last send message who will be used to send the next one.

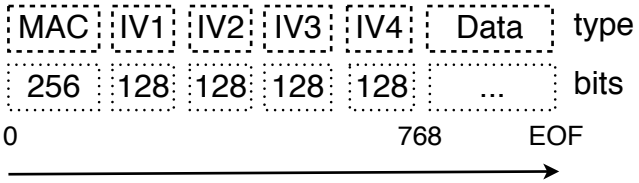
For security reasons every message with desynchronized IVs or an invalid MAC will be ignored.

Structure of the first message send:



MAC = SHA-256(IV3 + IV4 + Data)

Structure of the others messages :



MAC = SHA-256(IV1 + IV2 + IV3 + IV4 + Data)

Message

Messages aren't case sensitive and the lines returns are defined by the sequence **CR LF** (US-ASCII encoding defined by ANSI X3.4-1986) :

CR = <US-ASCII CR, carriage return (13)>
LF = <US-ASCII LF, linefeed (10)>

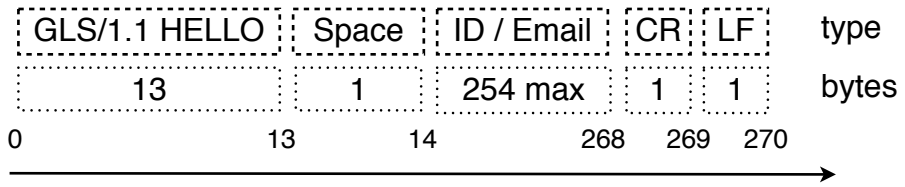
Hello Message

The Hello message structure is :

GLS/1.1 HELLO [My ID]

This message is send the first time in plaintext and a second time encrypted. They both must be verified. [My ID] represent the user's id, example :

GLS/1.1 HELLO user@domain.com
GLS/1.1 HELLO 5335216



The maximal hello message size is 270 bytes with 254 bytes for the user's id. All bigger message must be truncate. The minimal size is 17 bytes with 1 bytes for the user's id.

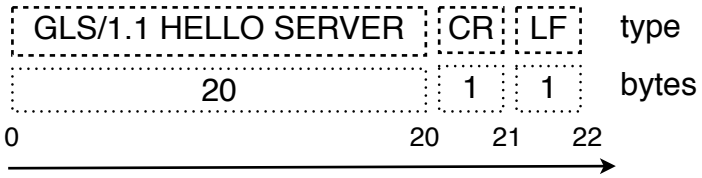
HelloServer Message

The server's response if the authentication was successful is always encrypted :

GLS/1.1 HELLO SERVER

Example :

GLS/1.1 HELLO SERVER



The Hello Server message size is 22 bytes.

Register Message

The Register message is send two times, the first one in plaintext without information :

GLS/1.1 REGISTER

The second time with the user's register informations. This message is always fully encrypted with the RSA public key from the server's X.509 certificate :

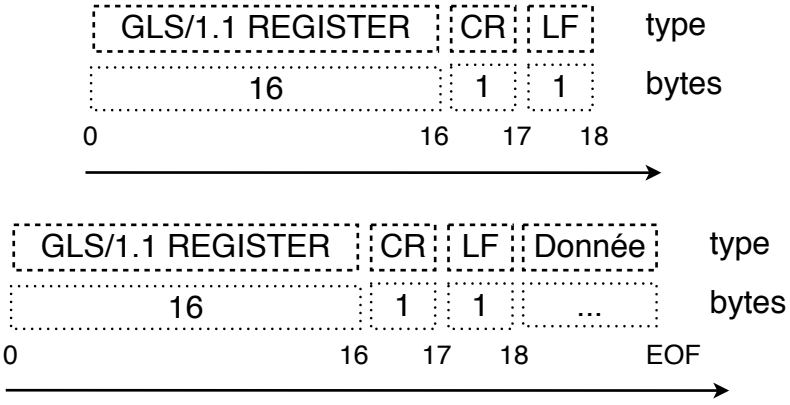
GLS/1.1 REGISTER
[Information]

Example :

GLS/1.1 REGISTER

Example 2 :

GLS/1.1 REGISTER
<?xml version="1.0" encoding="UTF-8" ?>
...



RegisterServer Message

The Register Server message is send two times, always in plaintext. The first message caries the server certificate (PEM format) :

GLS/1.1 REGISTER SERVER
[Certificat PEM base 64]

The second message is an acknowledgment of receipt that the informations have been received :

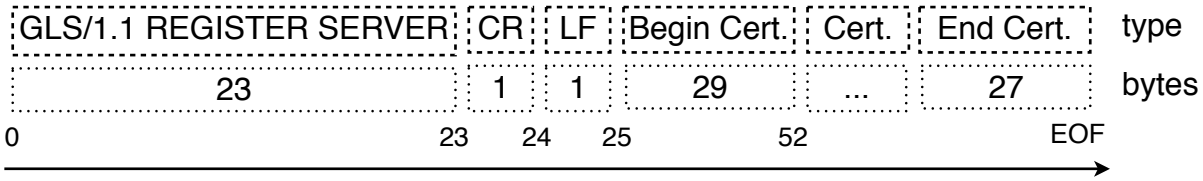
GLS/1.1 REGISTER SERVER OK

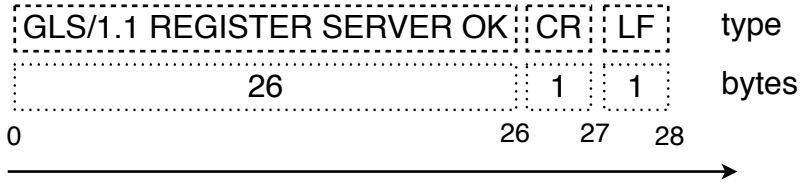
Example :

GLS/1.1 REGISTER SERVER
-----BEGIN CERTIFICATE-----
bGUuY29tMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC7n4lmsms4AQBFbmbd
2sdg6MJBlpAUU7kSSIfZMYBQ/yw22K2ZlHWCQuYnp1uqnZIEyYrQW2aoz2h6NWgI
QhvRWyEBEvATWLJqAzf9I5XiGm6fOj4ByzUKyyGTgx9aVOILwgGX/2qJzwSNKifL
7h4x0StHQfc6SNaS8kz+lqgaxwIDAQABo4HqMIHnMAkGA1UdEwQCMAAwKAYJYIZI
AYb4QgENBBsWGVNpZ25lZCBieSBFYW1vbm4ncyBPd24gQ0EwHQYDVR0OBBYEFKyk
ZjA2efblfWflOTY78K/FwgH/MIGQBgNVHSMEgYgwgYWAFLu4MXYKHe9tvI6RISWn
cPuXWNmjoWqkaDBmMQswCQYDVQQGEwJJRTEXMBUGA1UEChMOVGhlIE1jR29uaWds
ZXMxGTAXBgNVBAMTEFRoZSBNY0dvbmlnbGUgQ0ExIzAhBgkqhkiG9w0BCQEWFGVh
bW9ubkBib2dwZW9wbGUuY29tg
-----END CERTIFICATE-----

Example 2 :

GLS/1.1 REGISTER SERVER OK





Error Message

Errors message are always send in plaintext and the connexion is closed after :

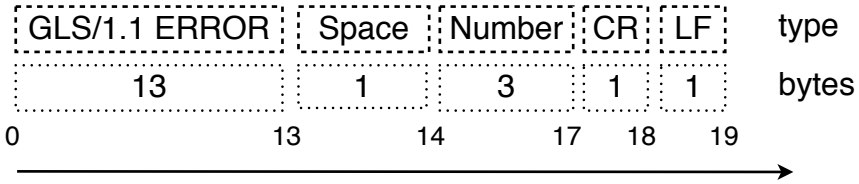
GLS/1.1 ERROR [number]

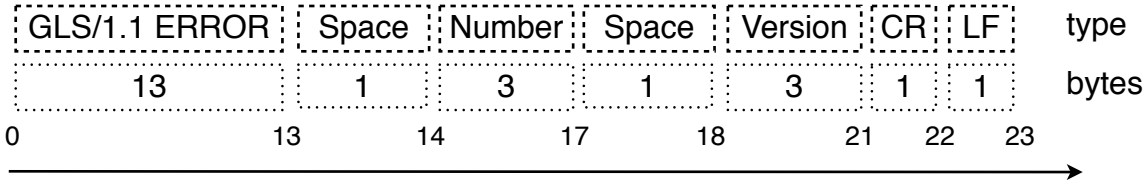
If the server doesn't support the GLS version it send back an 200 error message with [version] the minimal version needed to establish a connexion :

GLS/1.1 ERROR 200 [version]

Example :

GLS/1.1 ERROR 200 1.1





Version Number

The version number is written on 3 bytes, the first one for the major version, the second one for the dot and the last one for the minor version.

Error Number

Do not confuse the errors numbers from the section entitled « Using it ! » who are the library's errors codes and the following ones who are the GLS protocol's errors codes. If you are just using the library you will not work with the following ones.

Number	Description	Example
100	Impossible to read the encrypted HELLO message, authentication failed.	GLS/1.1 ERROR 100
101	Bad certificate, server authentication failed	GLS/1.1 ERROR 101
102	Missing certificate	GLS/1.1 ERROR 102
200	Unsupported GLS version	GLS/1.1 ERROR 200 1.1
300	Impossible to read the encrypted message	GLS/1.1 ERROR 300
301	Impossible to read the encrypted REGISTER message	GLS/1.1 ERROR 301
302	Impossible to read the encrypted HELLO SERVER message	GLS/1.1 ERROR 302
400	Unreadable message	GLS/1.1 ERROR 400
401	Unreadable HELLO message	GLS/1.1 ERROR 401
402	Unreadable HELLO SERVER message	GLS/1.1 ERROR 402
403	Unreadable REGISTER message	GLS/1.1 ERROR 403
404	Unreadable REGISTER SERVER message	GLS/1.1 ERROR 404
500	Internal error	GLS/1.1 ERROR 500

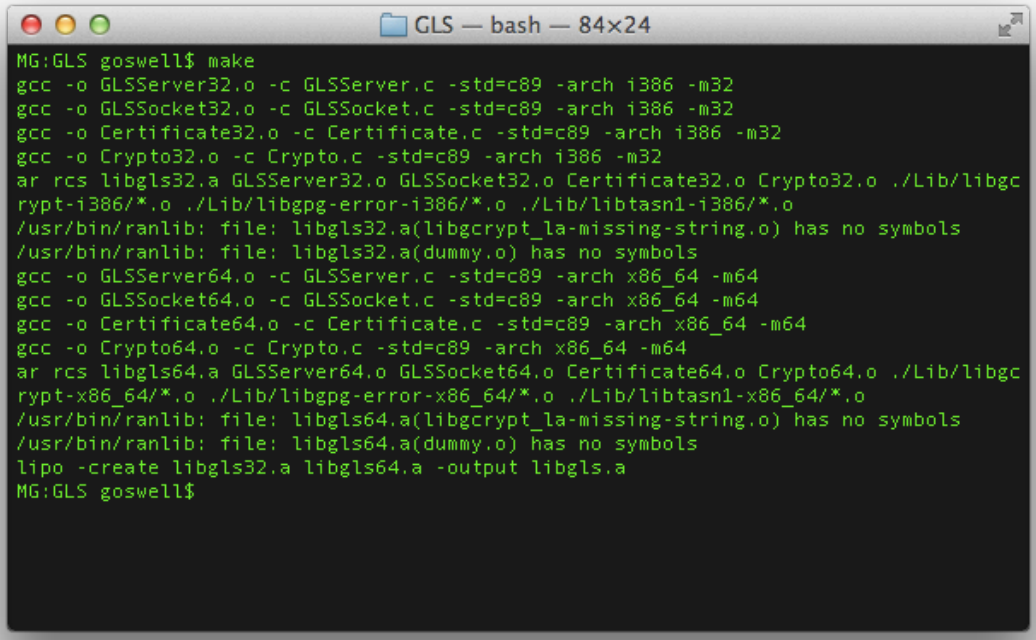
Using it !

Compilation

There is a lot of work to do on the makefile, right now it only compile on OS X Lion. Some update will be made on further releases to make it cross-platform, see « what's next ? » part for more informations. If you are on a different operative system you will have to compile the library manually. The source code works fine under Linux but Windows hasn't been tested yet. The library and all the dependent libraries are C89 compliant.

There is a pre-compiled static library on the Git repository for OS X Lion 32/64 bits. If you don't want to compile it just download libgls.a and libgls.h. Include these files in your project and you are good to go ! This package already contains all the require libraries for libgls to work, no need to do extra stuff.

To compile the library under OS X Lion, open a Terminal, go to the libgls directory and do a make :



You can now copy the libgls.a and libgls.h in your project. You can also remove all the temporary file using **make clean**. This created a static library with all the dependent libraries packaged into it.

Compiling It Manually

If you are on another operative system or you want to compile a dynamic library you will have to do it manually. This part of the documentation explain how to do it.

Compiling a static library

Download the source code from GitHub and remove all the content of the Lib directory. Create 3 directory in it : 'libtasn1', 'libgcrypt' and 'libgpg-error'.

Download libgpg-error from GNU's website and decompress the archive. Under a terminal go to the libgpg-error directory you just decompress and do :

```
$ cd libgpg-error-1.10
$ ./configure --enable-static=yes --enable-shared=no
$ make
$ cd src/
$ cp libgpg_error*.o /path/to/libgls/Lib/libgpg-error/
$ cd ..
$ mv src/ bin
```

Download libgcrypt from GNU's website and decompress the archive. Go to the libgcrypt directory you just decompress and do :

```
$ cd libgcrypt-1.5.0
$ ./configure --with-gpg-error-prefix=/path/to/libgpg-error-1.10/ --enable-
static=yes --enable-shared=no
$ make
$ cd src/.libs
$ ar x libgcrypt.a
$ cp *.o /path/to/libgls/Lib/libgcrypt/
```

Download libtasn1 from GNU's website and decompress the archive. Go to the libtasn1 directory you just decompress and do :

```
$ cd libtasn1-2.11
$ ./configure --enable-static=yes --enable-shared=no
$ make
$ cd lib/gllib/
$ cp *.o /path/to/libgls/Lib/libtasn1/
$ cd ..
$ cp *.o /path/to/libgls/Lib/libtasn1/
```

Open the libgls.h file and comment / uncomment the operative system in relation with the one you are using :

```
/*
 *
 *  CONFIGURATION
 *
 */

/*
 * Choose the OS for the compilation
 * it will be implemented in the make file on a next release
 * works fine on Linux and OSX, I didn't try it on windows.
 */

/* #define linux */
/* #define win32 */
#define osx
```

Go to the libgls source directory and do :

```
 $ cd /path/to/libgls/
 $ gcc -c GLSServer.c -o GLSServer.o
 $ gcc -c GLSSocket.c -o GLSSocket.o
 $ gcc -c Crypto.c -o Crypto.o
 $ gcc -c Certificate.c -o Certificate.o
 $ ar rcs libgls.a GLSServer.o GLSSocket.o Certificate.o Crypto.o ./Lib/
libgcrypt/*.o ./Lib/libgpg-error/*.o ./Lib/libtasn1/*.o
```

You can now copy the libgls.a and libgls.h in your project. This created a static library with all the dependent libraries packages into it.

Compiling a dynamic library

Download and install the libraries libgpg-error, libgcrypt and libtasn1. You just need to do a basic configure && make && make install. You will also need the root privileged to do the last one. If you're having trouble look at the libraries's documentations.

Once all is installed, download the libgls source code from GitHub, open a Terminal and go into the source's directory :

```
$ cd /path/to/libgls/
$ gcc -fPIC -c GLSServer.c -o GLSServer.o
$ gcc -fPIC -c GLSSocket.c -o GLSSocket.o
$ gcc -fPIC -c Crypto.c -o Crypto.o
$ gcc -fPIC -c Certificate.c -o Certificate.o
```

If you are creating a dynamic library for OS X use this command :

```
 $ gcc -dynamiclib -Wl,-headerpad_max_install_names,-undefined,dynamic_lookup,-
compatibility_version,1.0,-current_version,1.0,-install_name,/usr/local/lib/
libgls.1.dylib -o libgls.1.dylib GLSServer.o GLSSocket.o Certificate.o
Crypto.o /usr/local/lib/libgcrypt.dylib /usr/local/lib/libgpg-error.dylib /usr/
local/lib/libtasn1.dylib
```

Or if you are under linux use this one :

```
 $ gcc -shared -Wl,-soname,libgls.so.1 -o libgls.so.1 GLSServer.o GLSSocket.o
Certificate.o Crypto.o /usr/local/lib/libgcrypt.so /usr/local/lib/libgpg-
error.so /usr/local/lib/libtasn1.so
```

Don’t forget to copy the dynamic libgls into /usr/local/lib/ and the libgls.h file into /usr/local/include/. Now you can use the GLS library in your project just by adding :

```
#include <libgls.h>
```

Simple Connexion

```
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

    /* Initialize the socket */
    GLSSock* myConnexion = GLSSocket();

    /* Set the user's ID */
    setUserId(myConnexion, "myUserId");

    /* Add the user's password */
    addKey(myConnexion, "myPassword", 0);

    /* Connect to the server, you can use an IP or a domain name */
    connexion(myConnexion, "www.server.com", "443");

    /* You are now connected to the server using the GLS Protocol
    you can send and receive message with the function gslSend() and glsRecv() */

    /* Sending a message */
    byte *myMessage = "this is a message";
    glsSend(myConnexion, myMessage, strlen(myMessage));
```

```c
    /* Receiving a message, this is a blocking function */
    byte *anotherMessage = 0;
    int sizeMessage = glsRecv(myConnexion, &anotherMessage);

    /* You are responsible for deallocating the received message */
    free(anotherMessage);

    /* Close the connexion and free the GLS Socket */
    freeGLSSocket(myConnexion);

    return 0;

}
```

Simple Server

```c
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

    /* Allocate the server */
   GLSServerSock* myServer = GLSServer();

   /* Initialize the server with 10 waiting queue on the port 443 */
   initServer(myServer, "443", 10, 0);

   /* Wait for a client */
   GLSSock *myClient = 0;
   waitForClient(myServer, &myClient);

   /* Once you have a connexion, you will have to retrieve
    the user password from your database */

   /* Get the user's ID, this allocate new memory for the char */
   char *userID = 0;
   getUserId(myClient, &userID);

   /*
    * Some function to retrieve the user's password from the database
    */

   /* Free the userID once done with it */
   free(userID);

   /* Set the password, in this case is a SHA-512 so we set isSha = 1 */
   /* If your server uses a different hashing system, add it as a password
    and don't forget to also hash the user's password on the client */
   addKey(myClient, "752c14ea195c4...60bac3c3b789697", 1);

   /* Finish the handshake */
   finishHandShake(myClient);

   /* You are now connected with the client using the GLS Protocol
    you can send and receive message with the function gslSend() and glsRecv() */

   /* Sending a message */
   byte *myMessage = "this is a message";
   glsSend(myClient, myMessage, strlen(myMessage));
```

```
    /* Receiving a message, this is a blocking function */
    byte *anotherMessage = 0;
    int sizeMessage = glsRecv(myClient, &anotherMessage);

    /* You are responsible for deallocating the received message */
    free(anotherMessage);

    /* Close the connexion and free the GLS Socket */
    freeGLSSocket(myClient);

    /* Close the server and free the GLS Socket Server */
    freeGLSServer(myServer);

    return 0;

}
```

Register An User

```
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

    /* Initialize the socket */
    GLSSock* myConnexion = GLSSocket();

    /* Add the root certificate, be careful the current
     library only support RSA + SHA1 certificates */
    addRootCertificateFromFile(myConnexion, "./ca.crt");

    /* If you need to revocate some certificate
     add their ID to the CRL */
    addToCrl(myConnexion, "009D61A449B6BF4539");
    addToCrl(myConnexion, "00F7524FE8D6780e26");

    /* Create the register message */
    byte *message = "user's information to register";

    /* Send the register message to the server */
    sendRegister(myConnexion, "www.server.com", "443", message, strlen(message));

    /* Free the GLS Socket */
    freeGLSSocket(myConnexion);

    return 0;

}
```

Register An User (Server Side)

```c
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

   /* Allocate the server */
   GLSServerSock* myServer = GLSServer();

   /* Initialize the server with 10 waiting queue on the port 443 */
   initServer(myServer, "443", 10, 0);

   /* Add the server's certificates, be careful the current
    library only support RSA + SHA1 certificates */
   addServerCertificateFromFile(myServer, "./publicCert.crt", "./privateKey.key");

   /* Wait for a client */
   GLSSock *myClient = 0;
   waitForClient(myServer, &myClient);

   /* Get the register message from the client */
   byte *message = 0;
   int sizeMessage = getRegisterMessage(myClient, &message);

   /* You are responsible for deallocating the received message */
   free(message);

   /* Free the GLS Socket */
   freeGLSSocket(myClient);

   /* Close the server and free the GLS Socket Server */
   freeGLSServer(myServer);

    return 0;

}
```

Differentiate Connexions

```c
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

/* Allocate the server */
   GLSServerSock* myServer = GLSServer();

   /* Initialize the server with 10 waiting queue on the port 443 */
   initServer(myServer, "443", 10, 0);

   /* Add the server's certificates */
   addServerCertificateFromFile(myServer, "./publicCert.crt", "./PrivateKey.key");

   /* Wait for a client */
   GLSSock *myClient = 0;
   waitForClient(myServer, &myClient);
```

```
    /* Differentiate a standard from a register connexion */
    if(getTypeConnexion(myClient) == GLS_CONNEXION_STANDARD) {

        /* Handle a standard connexion with the client */

    }
    else if(getTypeConnexion(myClient) == GLS_CONNEXION_REGISTER) {

        /* Handle a register connexion with the client */

    }
    else {

        /* Error */

    }

    /* Free the GLS Socket */
    freeGLSSocket(myClient);

    /* Close the server and free the GLS Socket Server */
    freeGLSServer(myServer);

     return 0;

}
```

Handling Errors

```
#include <stdio.h>
#include "libgls.h"

int main (int argc, const char * argv[])
{

        /* Initialize the socket */
        GLSSock* myConnexion = GLSSocket();
        if (myConnexion == NULL) return GLS_ERROR_NOMEM;

        /* Set the user's ID */
        int error = setUserId(myConnexion, "myUserId");
        if (error < 0) {

            /* Free memory */
            freeGLSSocket(myConnexion);

            /* Return the error */
            return error;

        }

        /* Add the user's password */
        error = addKey(myConnexion, "myPassword", 0);
        if (error < 0) {

            /* Free memory */
            freeGLSSocket(myConnexion);

            /* Return the error */
            return error;

        }
```

```
        /* Connect to the server, you can use an IP or a domain name */
        error = connexion(myConnexion, "www.server.com", "443");
        if (error < 0) {

            /* Free memory */
            freeGLSSocket(myConnexion);

            /* Return the error */
            return error;

        }

        /* You are now connected to the server using the GLS Protocol
         you can send and receive message with the function gslSend() and glsRecv() */

        /* Sending a message */
        byte *myMessage = "this is a message";
        error = glsSend(myConnexion, myMessage, strlen(myMessage));
        if (error < 0) {

            /* Free memory */
            freeGLSSocket(myConnexion);

            /* Return the error */
            return error;

        }


        /* Receiving a message, this is a blocking function */
        byte *anotherMessage = 0;
        int sizeMessage = glsRecv(myConnexion, &anotherMessage);
        if (sizeMessage <= 0) {

            /* Free memory */
            freeGLSSocket(myConnexion);
            if(anotherMessage != NULL) {
                free(anotherMessage);
                anotherMessage = 0;
            }

            /* Return the error */
            return error;

        }

        /* You are responsible for deallocating the received message */
        if(anotherMessage != NULL) {
            free(anotherMessage);
            anotherMessage = 0;
        }

        /* Close the connexion and free the GLS Socket */
        freeGLSSocket(myConnexion);

        return 0;

}
```

Working With Threads

```c
#include <stdio.h>
#include "libgls.h"
#include <pthread.h>

void* handleClient(void* myClient);

int main (int argc, const char * argv[])
{

   /* Allocate the server */
   GLSServerSock* myServer = GLSServer();

   /* Initialize the server with 10 waiting queue on the port 443 */
   initServer(myServer, "443", 10, 0);

   while (1) {

       /* Init variable */
       GLSSock *myClient = 0;
       pthread_t thread;

       /* Wait for a client */
       waitForClient(myServer, &myClient);

       /* Create thread */
       pthread_create(&thread, NULL, handleClient, (void*)myClient);

   }

   /* Close the server and free the GLS Socket Server */
   freeGLSServer(myServer);

}

void* handleClient(void* myClient) {

   /*
    * Work with the client
    */

   /* Don't forget to free the GLS Socket */
   freeGLSSocket(myClient);

   return NULL;

}
```

Using Others Languages

If you are using another language than C you can find interfaces in the source code directory. There is only a Python and Cocoa interface but others will be added on furthers releases.

If you are using the Python interface, don't forget to use the right libgls's name in the initialization function. If the library isn't in a standard include directory put the full path.

```
def initGLSLibrary():
    global libGlsInit
    global libgls
    global libc
    if libGlsInit == False:
        # Replace the libgsl's name with the one on your system
        libgls = CDLL("libgls.dylib")
        # The name of your Libc library
        libc = CDLL("libc.dylib")
        libGlsInit = True
```

Errors Numbers

Standard socket connection errors

GLS_ERROR_ACCES
For UNIX domain sockets, which are identified by pathname: Write permission is denied on the socket file, or search permission is denied for one of the directories in the path prefix.

GLS_ERROR_PERM
The user tried to connect to a broadcast address without having the socket broadcast flag enabled or the connection request failed because of a local firewall rule.

GLS_ERROR_ADDRINUSE
Local address is already in use.

GLS_ERROR_AFNOSUPPORT
The passed address didn't have the correct address family in its sa_family field.

GLS_ERROR_AGAIN
No more free local ports or insufficient entries in the routing cache. For AF_INET see the description of /proc/sys/net/ipv4/ip_local_port_range ip(7) for information on how to increase the number of local ports.

GLS_ERROR_ALREADY
The socket is nonblocking and a previous connection attempt has not yet been completed.

GLS_ERROR_BADF
The file descriptor is not a valid index in the descriptor table.

GLS_ERROR_CONNREFUSED
No-one listening on the remote address.

GLS_ERROR_FAULT
The socket structure address is outside the user's address space.

GLS_ERROR_INPROGRESS
The socket is nonblocking and the connection cannot be completed immediately. It is possible to select(2) or poll(2) for completion by selecting the socket for writing. After select(2) indicates writability, use getsockopt(2) to read the SO_ERROR option at level SOL_SOCKET to determine whether connect() completed successfully (SO_ERROR is zero) or unsuccessfully (SO_ERROR is one of the usual error codes listed here, explaining the reason for the failure).

GLS_ERROR_INTR
The system call was interrupted by a signal that was caught.

GLS_ERROR_ISCONN
The socket is already connected.

GLS_ERROR_NETUNREACH
Network is unreachable.

GLS_ERROR_NOTSOCK
The file descriptor is not associated with a socket.

GLS_ERROR_TIMEDOUT
Timeout while attempting connection. The server may be too busy to accept new connections. Note that for IP sockets the timeout may be very long when syncookies are enabled on the server.

GLS_ERROR_AI_ADDRFAMILY
The specified network host does not have any network addresses in the requested address family.

GLS_ERROR_AI_AGAIN
The name server returned a temporary failure indication. Try again later.

GLS_ERROR_AI_BADFLAGS
hints.ai_flags contains invalid flags; or, hints.ai_flags included AI_CANONNAME and name was NULL.

GLS_ERROR_AI_FAIL
The name server returned a permanent failure indication.

GLS_ERROR_AI_FAMILY
The requested address family is not supported.

GLS_ERROR_AI_MEMORY
Out of memory.

GLS_ERROR_AI_NODATA
The specified network host exists, but does not have any network addresses defined.

GLS_ERROR_AI_NONAME
The node or service is not known; or both node and service are NULL; or AI_NUMERICSERV was specified in hints.ai_flags and service was not a numeric port-number string.

GLS_ERROR_AI_SERVICE
The requested service is not available for the requested socket type. It may be available through another socket type. For example, this error could occur if service was "shell" (a service only available on stream sockets), and either hints.ai_protocol was IPPROTO_UDP, or hints.ai_socktype was SOCK_DGRAM; or the error could occur if service was not NULL, and hints.ai_socktype was SOCK_RAW (a socket type that does not support the concept of services).

GLS_ERROR_AI_SOCKTYPE
The requested socket type is not supported. This could occur, for example, if hints.ai_socktype and hints.ai_protocol are inconsistent (e.g., SOCK_DGRAM and IPPROTO_TCP, respectively).

GLS_ERROR_AI_SYSTEM
Other system error, check errno for details.

GLS_ERROR_WOULDBLOCK
The socket is marked nonblocking and the requested operation would block. POSIX. 1-2001 allows either error to be returned for this case, and does not require these constants to have the same value, so a portable application should check for both possibilities.

GLS_ERROR_CONNRESET
Connection reset by peer.

GLS_ERROR_DESTADDRREQ
The socket is not connection-mode, and no peer address is set.

GLS_ERROR_INVAL
Invalid argument passed.

GLS_ERROR_MSGSIZE
The socket type requires that message be sent atomically, and the size of the message to be sent made this impossible.

GLS_ERROR_NOBUFS
The output queue for a network interface was full. This generally indicates that the interface has stopped sending, but may be caused by transient congestion. (Normally, this does not occur in Linux. Packets are just silently dropped when a device queue overflows.)

GLS_ERROR_NOMEM
No memory available.

GLS_ERROR_NOTCONN
The socket is not connected, and no target has been given.

GLS_ERROR_OPNOTSUPP
Some bit in the flags argument is inappropriate for the socket type.

GLS_ERROR_PIPE
The local end has been shut down on a connection oriented socket. In this case the process will also receive a SIGPIPE unless MSG_NOSIGNAL is set.

GLS_ERROR_CONNABORTED
A connection has been aborted.

GLS_ERROR_MFILE
The per-process limit of open file descriptors has been reached.

GLS_ERROR_NFILE
The system limit on the total number of open files has been reached.

GLS_ERROR_PROTO
Protocol error.
In addition, Linux accept() may fail if:

GLS_ERROR_PERM
Firewall rules forbid connection.

GLS_ERROR_NOSR

GLS_ERROR_SOCKTNOSUPPORT

GLS_ERROR_PROTONOSUPPORT

GLS_ERROR_ADDRNOTAVAIL
The specified address is not available from the local machine.

GLS_ERROR_ISDIR
The address argument is a null pointer.

GLS_ERROR_IO
An I/O error occurred.

GLS_ERROR_LOOP
A loop exists in symbolic links encountered during resolution of the pathname in address.

GLS_ERROR_NAMETOOLONG
A component of a pathname exceeded {NAME_MAX} characters, or an entire pathname exceeded {PATH_MAX} characters.

GLS_ERROR_NOENT
A component of the pathname does not name an existing file or the pathname is an empty string.

GLS_ERROR_NOTDIR
A component of the path prefix of the pathname in address is not a directory.

GLS_ERROR_ROFS
The name would reside on a read-only file system.

GLS_ERROR_HOSTDOWN

GLS errors

GLS_ERROR_USERNOTCONF
The user is not configured.

GLS_ERROR_NOPASSWD
The user's password is not configured.

GLS_ERROR_UNKNOWN
Intern error from the conception of the library, if this happend please send us an email.

GLS_ERROR_NOMESSAGE
You're trying to send an empty message.

GLS_ERROR_CRYPTO
Error from the cryptographic library.

GLS_ERROR_MAC
Message corrupt.

GLS_ERROR_IVDESYNC
The cryptographic's initiations vectors are desynchronized, this means the connection had gone wrong (packets aren't in order) or someone is trying to attack you using replay method. You should close the socket and open a new connection.

GLS_ERROR_BADPASSWD
The login / password information aren't correct.

GLS_ERROR_BADSERVERCERT
The server certificate cannot be validated.

GLS_ERROR_VERSION
The GLS version of the library isn't compatible with the corespondent.

GLS_ERROR_TOMANYKEY
You can’t use more than 10 password on a socket.

GLS_ERROR_BADSIZEKEY
Bad SHA-512 string size.

GLS_ERROR_BASE64
Error trying to convert a base64 information to a byte information.

GLS_ERROR_ASN1
Error when using the library libtasn1.

GLS_ERROR_NOCERT
No certificate.

Library Functions

GLSSock* GLSSocket()

Allocate a GLS socket. GLSSocket() initialize libgcrypt, if your application use it too and you want secure memory don't forget to initialize the library from your application and add 16k of memory for GLS. By default GLSSocket() use secure memory, if you want more memory or want to disable it use GLSSocketSecure(). You are responsible for deallocating the socket with freeGLSSocket().

Return a pointer to the GLS socket or NULL.

GLSSock* GLSSocketSecure(const int secureMem, const int sizeMem)

Allocate a GLS socket. This function permit to configure the libgcrypt secure memory. You are responsible for deallocating the socket with freeGLSSocket().

int secureMem - Using secure memory or not (0 for false, 1 for true).
int sizeMem - The size of the secure memory in bytes.

Return a pointer to the GLS socket or NULL.

void freeGLSSocket(GLSSock* myGLSSocket)

Close the connexion and free the GLS socket.

GLSSock* myGLSSocket - The socket to free.

Return void.

int connexion(GLSSock* myGLSSocket, const char* address, const char* port)

Connect the socket to a GLS Server, you need to add an encryption key first with addKey() and an id with setUserId().

GLSSock* myGLSSocket - The socket used for the connexion.
const char* address - The server's address. You can use an IPV4 or a host name. IPV6 should work but hasn't been tested yet.
const char* port - The server's connexion port.

Return 0 for success, a negative number for an error.

int sendRegister(GLSSock* myGLSSocket, const char* address, const char* port, const byte* buffer, const int sizeBuffer)

Send an register message to a GLS Server. You need to add a root certificate first with addRootCertificate().

Return 0 for success, a negative number for an error.

int getRegisterMessage(GLSSock* myGLSSocket, byte** message)

A server side function. Get the register message send by sendRegister() on the client. You are responsible for deallocating message with free().

GLSSock* myGLSSocket - The socket to use.
byte** message - A non allocated pointer to receive the register message.

Return the message's size or a negative number for an error.

int glsSend(GLSSock* myGLSSocket, const byte* buffer, const int sizeBuffer)

Send a message using the secure connexion. You can use this function on a thread. GlsSend() works with bytes, so if you want to send something bigger like ints or floats be careful with the byte ordering.

GLSSock* myGLSSocket - The socket to use.
const byte* buffer - The message to send.
const int sizeBuffer - The message's size in bytes.

Return the message's size send or a negative number for an error.

int glsRecv(GLSSock* myGLSSocket, byte** buffer)

Wait for a message, you can use this function on a thread. You are responsible for deallocating the buffer with free(). This function is blocking, it will wait until a message is received.

GLSSock* myGLSSocket - The socket to use.
const byte** buffer - A non allocated pointer to receive the message.

Return the size of the received message or a negative number for an error.

int addKey(GLSSock* myGLSSocket, const char* key, int isSha)

Add user's password, you can have 10 different password. If the password is already in SHA-512, use the function with isSha = 1. The maximum password's length is 60 bytes. If you're using a different hash mechanism like md5, add them as a simple password. Don't forget to also hash the user's password on the client side before adding it. It's possible to change the passwords when the socket is connected, this will automatically change the encryption key and needs to be done on the server side too if you don't want to loose the connexion.

GLSSock* myGLSSocket - The socket to use.
const char* key - The key to add to the socket.
int isSha - If the key is a SHA-512 hash set it to 1, otherwise to 0.

Return 0 for success, a negative number for an error.

int clearKey(GLSSock* myGLSSocket)

Remove all the key added by addKey().

GLSSock* myGLSSocket - The socket to use.

Return 0 for success, a negative number for an error.

int getTypeConnexion(GLSSock* myGLSSocket)

Return the connexion's type (GLS_CONNEXION_STANDARD or GLS_CONNEXION_REGISTER).

GLSSock* myGLSSocket - The socket to use.

Return the connexion's type or a negative number for an error.

int getUserId(GLSSock* myGLSSocket, char** userId)

Get the user's id. You are responsible for deallocating userId.

GLSSock* myGLSSocket - The socket to use.
char** userId - A non allocated pointer to receive the userId.

Return the userId's size or a negative number for an error.

int setUserId(GLSSock* myGLSSocket, const char* userId)

Set the user's id.

GLSSock* myGLSSocket - The socket to use.
char* userId - A pointer to the user's ID char array.

Return 0 for success, a negative number for an error.

int finishHandShake(GLSSock* myGLSSocket)

A server side function. Finish the handshake for the connexion, you can after use glsRecv() and glsSend().

GLSSock* myGLSSocket - The socket to use.

Return 0 for success, a negative number for an error.

int addRootCertificate(GLSSock* myGLSSocket, const char* cert)

Add a root certificate for the Register connexion. PEM format. The current library only support certificates with RSA keys and SHA-1 signing.

GLSSock* myGLSSocket - The socket to use.
const char* cert - The certificate to add.

Return 0 for success, a negative number for an error.

int addRootCertificateFromFile(GLSSock* myGLSSocket, const char* certFile)

Add a root certificate from a file for the Register connexion. PEM format. The current library only support certificates with RSA keys and SHA-1 signing.

GLSSock* myGLSSocket - The socket to use.
const char* certFile - The certificate's path to add.

Return 0 for success, a negative number for an error.

int addToCrl(GLSSock* myGLSSocket, const char* serial)

Add a certificate's serial number to the CRL.

GLSSock* myGLSSocket - The socket to use.
const char* serial - The certificate's serial number.

Return 0 for success, a negative number for an error.

GLSServerSock* GLSServer()

GLSServer use libgcrypt, if your application use it too and you want secure memory don't forget to initialize the library from your application and add 16k of memory for GLS. By default GLSServer() use secure memory, if you want more memory or want to disable it use GLSServerSecure(). You are responsible for deallocating the socket with freeGLSServer().

Return a pointer to the Socket Server or NULL.

GLSServerSock* GLSServerSecure(const int secureMem, const int sizeMem)

Allocate a Socket Server. This function permit to configure the libgcrypt secure memory. You are responsible for deallocating the socket with freeGLSSocket().

int secureMem - Using secure memory or not (0 for false, 1 for true).
int sizeMem - The size of the secure memory in bytes.

Return a pointer to the Socket Server or NULL.

void freeGLSServer(GLSServerSock* myGLSServerSock)

Close the connexion and free the GLS server socket.

GLSServerSock* myGLSServerSock - The server's socket to free.

Return void.

int initServer(GLSServerSock* myGLSServerSock, const char *port, const int waitQueue, const int isReuse)

Initialize the server for listening on a port.

GLSServerSock* myGLSServerSock - The server's socket to use.
const char *port - The port to listen on.
const int waitQueue - The size of the waiting list.
const int isReuse - Force the socket to listen on a TIME_OUT port.

Return 0 for success, a negative number for an error.

int waitForClient(GLSServerSock* myGLSServerSock, GLSSock myClient)**

Wait for a connexion and allocate a GLSSock on myClient. You are responsible for deallocating the socket with freeGLSSocket().

GLSServerSock* myGLSServerSock - The server's socket to use.
GLSSock** myClient - A non allocated pointer to receive the GLS socket.

Return 0 for success, a negative number for an error.

int addServerCertificate(GLSServerSock* myGLSServerSock,
const char* publicCert, const char* privateKey)

Add the server certificate for the Register connexion. PEM format. The current library only support certificates with RSA keys and SHA-1 signing.

GLSServerSock* myGLSServerSock - The server's socket to use.
const char* publicCert - The server's public certificate.
const char* privateKey - The server's private certificate.

Return 0 for success, a negative number for an error.

int addServerCertificateFromFile(GLSServerSock* myGLSServerSock,
const char* publicCertFile,
const char* privateKeyFile)

Add the server certificate from a file for the Register connexion. PEM format. The current library only support certificates with RSA keys and SHA-1 signing.

GLSServerSock* myGLSServerSock - The server's socket to use.
const char* publicCert - The server's public certificate path.
const char* privateKey - The server's private certificate path.

Return 0 for success, a negative number for an error.

What's next ?

Simple / Double Encryption

Cascading encryption can be slow, it is always a compromise between security and speed. It will be added a simple mode encryption with one algorithm. This mode will be set on the client side. On the server's side it will be possible to force a double / simple encryption or to use an intelligent mode who will automatically detect witch one is used and configure the library in consequence.

Password Derivation

The GLS Protocol actually derive the encryptions keys from the password using a simple SHA-512 function. This cause all the message to be encrypted with the same key. In further release the derivation method will be improved using a pseudo-random function or a key derivation function to « randomize » the encryptions keys. This way all the message will have his own different encryption key.

CTR instead of CBC

CBC mode is actually used to encrypt the messages. This mode don't permit to thread the encryption and distribute the operations between the different processors. This really slow down the library. It will be replace by CTR permitting to encrypt different block at the same time allowing to speed up the encryption with multi-processor systems.

More Speed

Libgcrypt is one of the slowest cryptographic library in the open source world. It has been chosen for other reason like the C89 compliance. You can look at this article for more details on cryptographic library's speed. Further releases will bypass the library to directly use some slighter code of the encryption algorithms to gain more speed.

Multi-processor

The library will be able distribute the encryptions / decryptions operations between the different processors using CTR.

Deny Of Service

It will be added some mechanisms to prevent Deny Of Service on the server socket.

Makefile

The actual makefile only permit the compilation from OS X Lion using the pre-compiled dependent library's module. There is a lot of improvement to bring to it. It will be more cross-platform and allow the compilation of a static and shared library. No more pre-compiled module will be used instead the source code of the dependent library's will be furnish and automatically compiled by the makefile.

Multiple Root Certificate

To be able to change smoothly between different root certificate for security purpose, the library will handle multiple root certificate.

Full Security Check

The library has been debugged but it still need to pass some security test.

Apache & Firefox Module

The library was originally design for an application's use, specially the certificate system. But it would be nice to extend the scope of GLS to the web. In further release it will be develop an Apache and Firefox module for GLS.

Interface

Some interface with other languages have been develop but they are not implementing all the libgls's functionalities. They will be improved with time.

License

GNU GENERAL PUBLIC LICENSE

Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Lesser General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients'

exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
one line to give the program's name and an idea of what it does.
Copyright (C) yyyy  name of author

This program is free software; you can redistribute it and/or
modify it under the terms of the GNU General Public License
as published by the Free Software Foundation; either version 2
of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
GNU General Public License for more details.

You should have received a copy of the GNU General Public License
along with this program; if not, write to the Free Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301, USA.
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
Gnomovision version 69, Copyright (C) year name of author
Gnomovision comes with ABSOLUTELY NO WARRANTY; for details
type `show w'.  This is free software, and you are welcome
to redistribute it under certain conditions; type `show c'
for details.
```

The hypothetical commands `show w' and `show c' should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than `show w' and `show c'; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

```
Yoyodyne, Inc., hereby disclaims all copyright
interest in the program `Gnomovision'
(which makes passes at compilers) written
by James Hacker.

signature of Ty Coon, 1 April 1989
Ty Coon, President of Vice
```

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Lesser General Public License instead of this License.

GNU LESSER GENERAL PUBLIC LICENSE

Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts
 as the successor of the GNU Library Public License, version 2, hence
 the version number 2.1.]

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages--typically libraries--of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library.

To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.
b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.
c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.
d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful. (For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files in the Library will not necessarily be able to recompile the application to use the modified definitions.)
b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.
c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.
d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.
e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.
b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that

distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Libraries

If you develop a new library, and you want it to be of the greatest possible use to the public, we recommend making it free software that everyone can redistribute and change. You can do so by permitting redistribution under these terms (or, alternatively, under the terms of the ordinary General Public License).

To apply these terms, attach the following notices to the library. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

one line to give the library's name and an idea of what it does.
Copyright (C) *year name of author*

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This library is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public
License along with this library; if not, write to the Free
Software
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
02110-1301 USA
Also add information on how to contact you by electronic and paper mail.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the library, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in
the library `Frob' (a library for tweaking knobs) written
by James Random Hacker.

signature of Ty Coon, 1 April 1990
Ty Coon, President of Vice

That's all there is to it!