SSL certificates, MIDP certificates, eclipseME, J2ME and WTK.

So you want to write an SSL application on J2ME and you don't know how. I've just spent the whole night doing this so let me show you rather then try to crack your head open with this one.

First of all note this, a certificate is as the name suggests is there to certify and sign your application and it's connections over SSL, it is therefore pointless to make your own certificates and hope that everyone would trust your good intentions, for actual deployment of applications you'd need to buy a certificate from an authority such as Verisign, but for development and testing purposes your own certificate would do.

OK, I would assume you're using J2ME and java running on a linux machine, that you already have sun's WTK, eclipse with eclipseME and Java-6, don't ask me how to do it on any other configuration.

First thing, on my Debian, the Java installation was really messy, this meant that some parts of it were using the gcj as in "/usr/lib/jvm/java-gcj" and some parts we're using the sun's version on "/usr/lib/jvm/java-6-sun/", not to mention the Java-1.5 that is just sitting idle. To fix that run(as root) $***update-alternatives –all*** and set all of the Java things to one place, I used the Java-6-sun for all things, otherwise it uses different keytools (To generate keys) and it's a complete mess.

Once you've done that, run keytool, with your own options. For example
$keytool -genkey -alias company -keyalg RSA -validity 365

follow the instructions...
$ls -latr
look for a "*.keystore*" file this is what you'll use on a standard j2se application such as the server side, or any other application that isn't the emulator.
For example:
$java -Djavax.net.ssl.keyStore=<Keystore file> -Djavax.net.ssl.keyStorePassword=<Password> <Application>

Next run the ktoolbar on the *WTK<version>/bin* directory.
Go to file->utilities and choose "sign midlet".
Click on Import keypair from keystore.
Select all files (*.*) and give it the keystore file, if you've used the default keystore file name (as in this example) then you won't be able to see it, just go to the directory where it resides and type .keystore in the file name. Give it the password and all of the information it wants, leave manufacturer there
now if you run
mekeytool -list
in the *WTK/bin* you should be able to see your new keystore there

Next go to your j2me project in eclipse and
project->properties
from the side bar expand J2ME and select "Midlet Suite Signing".
Tick sign project, and select "external..."
navigate to /home/<yourusername>/j2mewtk/2.5.2/appdb/keystore.ks give it some alias and select save password as part of project (I think you don't have to give that but it's easier that way) .
Verify settings,apply & OK

then go to run->run configuration, expand wireless toolkit emulation and select your run configuration. Then under the emulation tag you should change security domain to manufacturer(or whatever you chose in the ktoolbat->utils->sign midlet part
that's it, you now have a certificate you can test your apps with.

When it's time for you to buy a certificate, make sure it'll be compatible with as many different devices as you can, I hear Verisign are a good one.
Hope I didn't forget anything, let me know of any improvements I should do to this document.

good luck.

See : http://stackoverflow.com/questions/1383771/how-do-you-sign-a-java-midlet/1383823
for some more information.