**Multiplayer Assembly Package  
Tutorials  
Playing it safe with Cryptography**

So, now that you have this package installed, you have a big grouping of security functions and network transmission code. The question is, when do I use these pieces of code outside the current setup? And is the setup secured perfectly?

Well, sadly, there is no such thing as a perfect secured piece of code, but this quick guide will ensure you are following the route the most securely when you need to.

First, open the file cryptoPackage.cpp. You will notice the generic AESFileKey() is in place. This really isn’t the best route to go with file security as the smallest file could turn into a disaster. Instead, you might want to replace that portion of the package with something more along the lines of “file based key” where you have an array of file keys depending on the file name, ect. If you were really crafty, you could combine a general key with the file path as the key, and be even more evil to tricky hackers.

Next, we will discuss message transmission. As seen in the code tutorial server based messages, the transmission is heavily secured using RSA. I highly recommend going this route for pretty much everything that has even the slightest bit of modifiable information. Don’t take this to the extreme however, or you will kiss your performance goodbye on encryption cycles. The news is an example of something that doesn’t need to carry encryption as it is almost exactly like opening a web page.

All in all, I recommend keeping a simple checklist on hand:

[ ] Does it Contain Modifiable Information?

[ ] Does it affect other players?

[ ] Do I want people to see this info?

[ ] Am I giving access to anything by means of this code?

If you checked off any one of these items, you should probably secure it.

Recommendations:

* AES Encryption
* RSA Encryption (Requires the Server)
* Both (Requires the Server)
* SHA1, Whirlpool, PGD Hash (Comparing two strings, one stored)