HSEMM DESIGN NOTES

RUPERT STANLEY

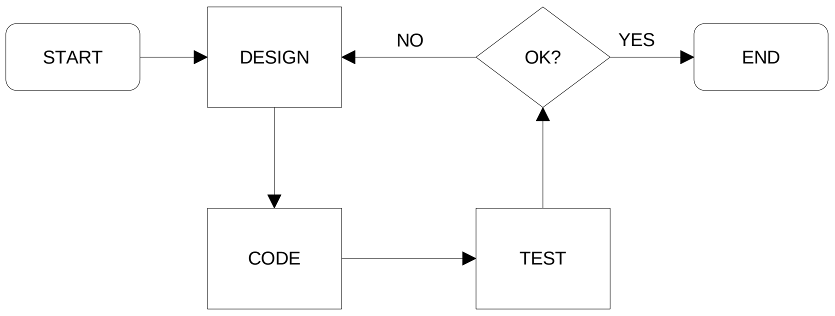
ROSS SYSTEMS INTERNATIONAL

HSEMM came about because I had some major problems in developing applications using bespoke HSM firmware, these two problems came in the form:

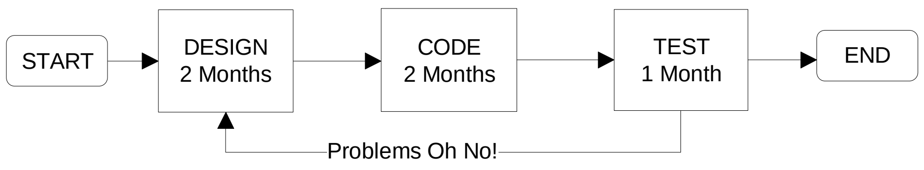
1. It was very difficult to specify and test new cryptographic functions in the HSM
2. It took absolutely ages to complete the project

Let me explain.

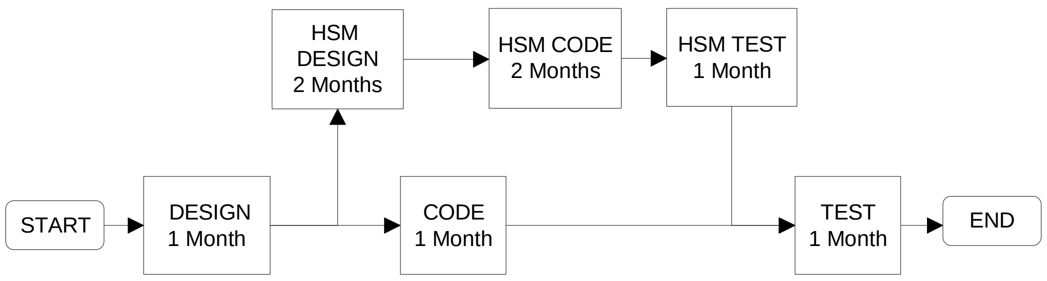
In a normal project, iterative development is possible because the time from error detection to correction can be very short, minutes or hours at worst.



However, with HSMs the delays are very much longer



Compound this with applications software development



Typical project duration on critical path is 7 months, this is not good!

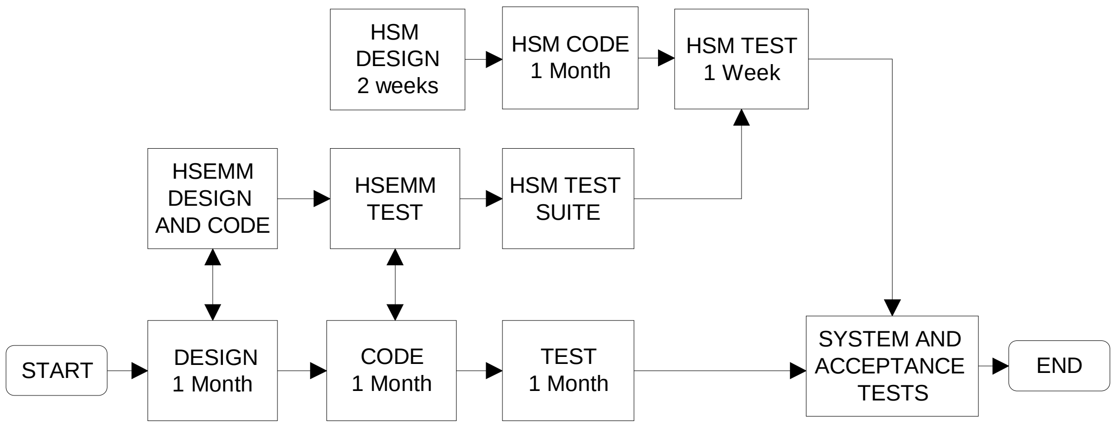
The reason why this is so is because suppliers of HSM firmware want each stage to be complete before progressing with the next, so:

1. The design is reworked again and again before being approved.
2. The HSM coding takes place using a small very skilled team who can only do so much in a given time.
3. Finally when you receive the firmware the testing is very protracted because it takes a considerable amount of time to write the test scenarios, run them and finally check them by hand.

This all changes when you use an HSM Emulator.

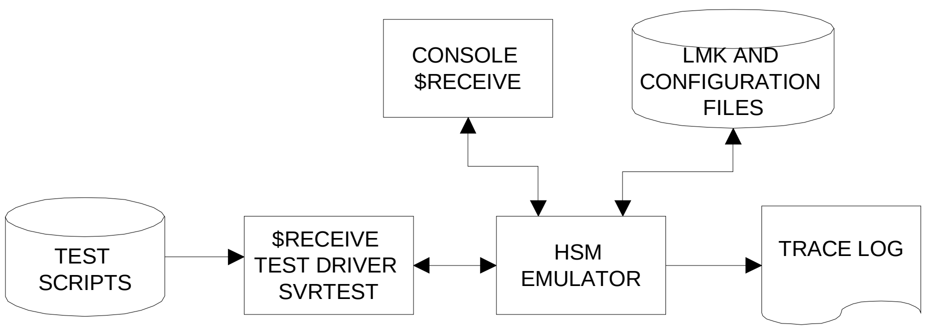
1. The firmware design is performed in conjunction with your project team by us and prototyped for the HSM emulator on the fly, so that project coding can commence immediately the prototype HSM functionality has been approved, by testing using the scripting tools, using iterative development techniques.
2. The HSM firmware specification can be written and approved with the minimum delay since it is known exactly what is wanted and the corresponding C code can be submitted with it. This can be done in parallel with the application development, so that any problems found during development can be addressed in the final approved specification, once again iterative development can be used..
3. The development of the HSM code is considerably expedited by the HSM supplier having the C code, and detailed system testing can be performed in parallel with this coding, including the writing of an HSM test suite to test the firmware supplied.
4. Testing the HSM is a quick affair since the HSM can be tested by parallel running with the HSM Emulator.
5. The final tests against the HSM can be simply final systems integration and acceptance tests.

Lets see how this looks in the following diagram:



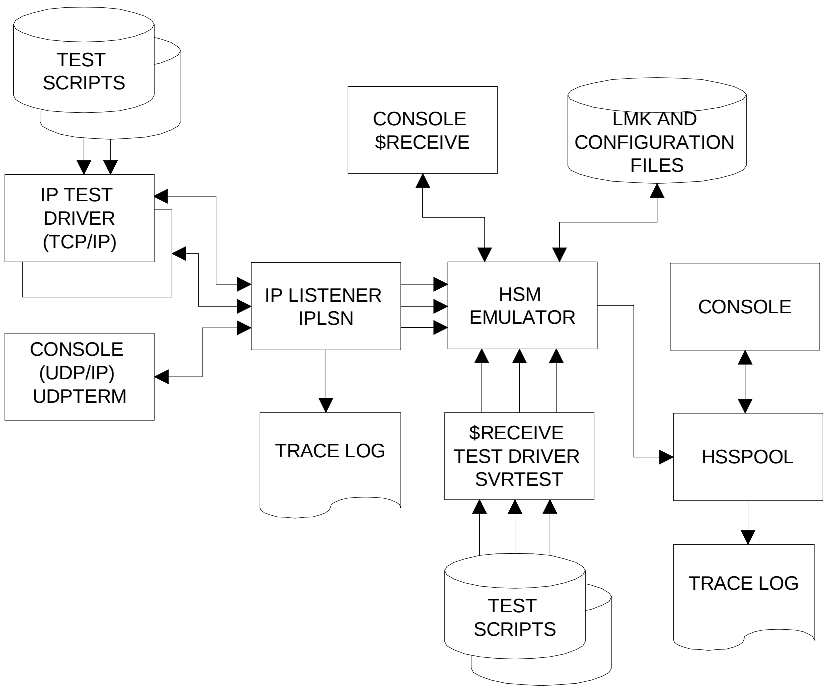
Total project duration is about 3.5 months, half the time of the conventional approach, and all the components have been thoroughly quality tested with audit traces.

Firstly lets inspect the HSEMM design and code phase. In this phase the HSM commands are designed and entered into the script file, the prototype HSM commands are coded into the HSM Emulator and the script files run in the HSEMM system.



Note. Both Console and Host commands can be prototyped at this phase, and refined using a process of iterative development.

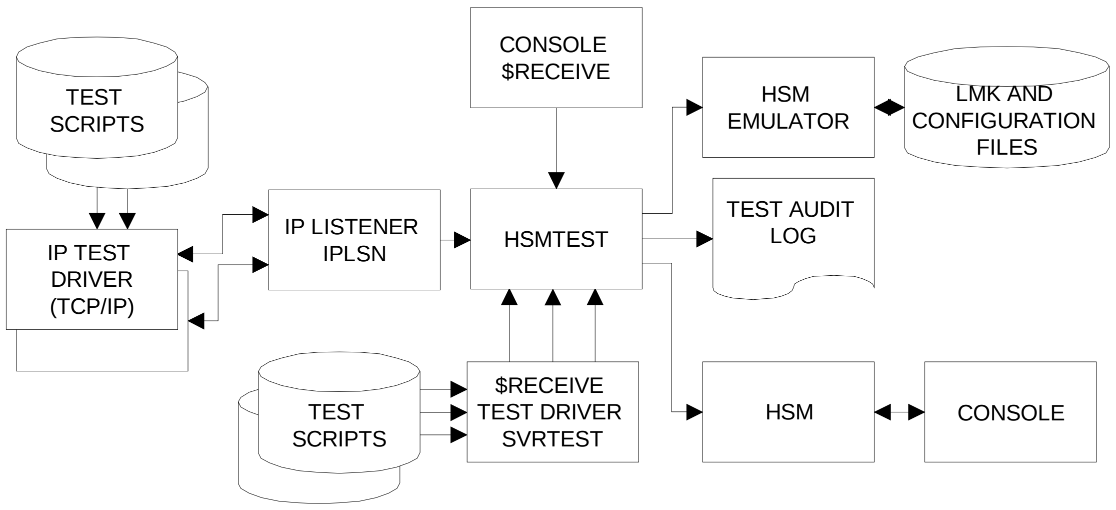
Lets have look now at the HSEMM Test phase, because this is the phase is which all the HSEMM elements come into play.



Note. It is possible to test every aspect of the final HSM including the multithreading element at the same time as providing a full cryptographic test.

This is also the same system which is used for the program testing except that SVRTEST and/or IPTEST are replaced by the application.

The testing of the HSM is performed using the module HSMTEST, which contains a programmed section for the purpose of resolving random keys. The HSM Emulator contains a special command for loading ‘random’ key sets. This is to ensure that the output from both the HSM and emulator are consistent under all circumstances.



1. HSMTEST has both a $RECEIVE and TCP/IP client interface for testing the various types of HSM.
2. The Test audit log can use HSSPOOL for a permanent record
3. The IPTEST and SVRTEST drivers can be replaced by your application to enable you to perform an audited online test of your HSM.

Conclusion.

HSEMM has presented many challenges during its development, and further refinement, from the initial desire to break the mould of HSM based application development using new firmware to its final realization as a working product.

On demonstration, it has been remarked that not only does it do what it was intended for, that is:

To speed, simplify and improve the quality of HSM firmware development and the projects dependant on it

But also:

It breaks the iron grip of the group managing the production HSM keys on these projects, by enabling access of the whole project group to a virtual HSM which they can configure at will, without compromising corporate security, for instance it is only possible to load master keys onto the HSM emulator from files and not Cards and also there is a difference in the way in which the LMKs are calculated which means that LMKs can never be compromised even if a developer obtains the LMK Initial key values.

It provides great insight into the functioning and operation of HSMs and is a valuable educational tool.

Finally, and probably for you most importantly; your cryptographic functions are fully enabled, secure and secret, because we never allow any third party, except your HSM supplier, access to any part of your functionality either in the form of source or object code or documentation.

In future, the product will be ported to NonStop OSS/UNIX/LINUX and to the Microsoft Windows Environments.

Rupert Stanley

Mistley

25 April, 2006

Ross Systems International Ltd.
Tel. +44(0)1206-392923.
e-Mail info@rsi-ns.com