Ross Systems International

White Paper

High Performance Solutions for Thales HSMs Linked to HP NonStop (Tandem) Systems

by

Rupert Stanley

Ross Systems International Limited.
The Hollies
18 New Road
Mistley
Manningtree
Essex CO11 2AG
Tel. +44-1206-392923
Email. info@rsi-ns.com

Abstract.

Host Security Modules (HSMs) are extensively used in the Financial Service Industries to provide a secure, trusted environment to perform sensitive operations, such as PIN / signature verification, and message authentication.

Over the last ten years the even increasing use of chip and PIN and other ATM transactions has had the result that both the volume and crypto mix of HSM transactions has increased. The importance of this elevated activity has resulted in the need for better management, control, and fault tolerance of the HSM devices and the networks and applications they co-exist with.

In short these electronic banking services have moved from the periphery to being central to any financial institutions offering to its clients, with peak loads of many thousands of transactions per second, and as such the reputational damage which will result from the failure of any part of the service is huge, with a prospect of equally significant litigation by injured clients.

This paper will address the ways for improving the resilience of HP NonStop Tandem Systems using Thales (Racal) HSMs together with steps to be taken to reduce the cost per transaction by improving fault tolerance, throughput, management and reporting of the service, without significantly increasing individual transaction processing times. Highlighting any constraining factors in the proposed solutions and proposing methods to resolve any conflicts encountered.

Introduction.

This white paper is structured so that firstly the commercial imperatives of each aspect of the technology is discussed and then the ways in which the situation can be improved by levering appropriate techniques and suitable improvements is explored.

The current situation is that in order to manage peak load, without degradation of performance, a typical card processing institution will need to have many Thales 7000/8000 HSMs, to handle a peak load will of about 10,000 tps.

This peak load has shown a consistent growth over the last 5 years of about 15% per annum and shows no sign of decreasing. Thus requiring an addition of several HSMs and their associated infrastructure per year, to maintain service levels.

A high performance HSM (Thales HSM8-EH) costs about £30,000 plus an extra 10% for installation and maintenance and thus it is not uncommon for about £100,000 per annum to be expended by these institutions for new cryptographic equipment and infrastructure.

However, this figure does not factor in the ever increasing risk of failure of part or all of the HSM service, owing to this ad hoc and relatively unplanned growth, in the fact that the more complex a service is the more likely it is to fail unless steps have been taken to plan for failure of one or more components and automatic recovery from any loss of service.

The resulting situation is that the total capital cost based on risk plus expenditure is rising exponentially and has already reached such a level that several institutions have been badly impacted by electronic banking system failures, and it is only a matter of time before this becomes commonplace with all the losses involved. It is also to be noted that such unplanned, but nevertheless foreseeable, service interruptions are in contravention of the Basel and PCI DSS recommendations and regulations.

There is therefore a need to plan to provide a resilient service which at the same time makes the very best possible use of the expensive infrastructure (devices, network and host services).

The problems can be divided into the following areas.

HSM Performance

The speed at which an HSM can perform transactions is of vital commercial importance to any user. Since if the throughput of an HSM can be doubled then the number of HSMs required will be halved with the corresponding cost savings in the commissioning of new HSM to meet an institutions demands, which can be in the range of £100,000 p.a. for a major user.

The Thales HSM8-EH HSM, which is quoted as having a maximum transaction rate of 800 tps, uses a high performance CPU Chip and DES Co-Processor which can theoretically perform many thousands of Triple DES PIN Translates per second, based on the research figures into the number of cycles required per PIN Translate.

The difference between, the actual and the theoretical performance is owing to the overheads of:

1. The HSMs operating system
2. The need to traverse 6 OSI stack layers in both the Host Computer and the HSM
3. The need to interpret the HSM command

The ways in which this performance can be improved, in a flexible manner to meet current and future needs, will be examined highlighting the technical and commercial challenges.

HSM Utilisation

The utilisation of the HSMs individually within a group of HSMs is commercially important since if the load is not shared equally amongst the available HSMs there will be a requirement for an increased number of HSMs so that no single HSM becomes a bottleneck to the flow of data through the electronic banking product set. This end result of this imbalance is the additional cost of these (unnecessary) HSMs and their maintenance.

The key factors in avoiding the situation, where at any given instant certain HSMs will be completely committed whilst other are under-utilised, lies in the way in which HSM processing power is allocated to applications.

In most environments where HSMs are used the selection of the HSM to perform any given task is either static or session based, if a dynamic HSM allocation scheme is used.

However, there are certain HSM transactions, such as RSA key pair generation, which take a significant time, up to tens of seconds, which cannot be mixed with transactions such as PIN Translation and Verification which need sub-second response times. Also the processing capabilities of Thales 7000 and 8000 HSMs differ widely and must also be taken into account.

The methods which can be used to achieve HSM usage optimisation will be examined together with an analysis of the connectivity issues concerning the linking of Thales HSMs to HP NonStop Systems and the associated matters concerning the building of a fault tolerant service which is also adaptable to meet future needs.

Management and Reporting

The management and reporting methods used in most of the current HSM based services seems either to have been completely ignored by most solution providers or added as an afterthought.

However, it is essential that this area is addressed in any high performance solution involving HSMs to enable the most cost effective planning for the deployment of HSMs and to provide the high levels of reliability needed by current electronic banking systems required to reduce the exposure of the financial institutions running them to either reputational and litigation losses.

Therefore the management and reporting component of HSM service design needs to be built in as an integral part of the service and will affect all the other components, with associated opportunities and challenges.

These will be addressed, and solutions proposed, based on the facilities available in HP NonStop (Tandem) systems, highlighting any constraining factors together with suggested methods to resolve these conflicts.

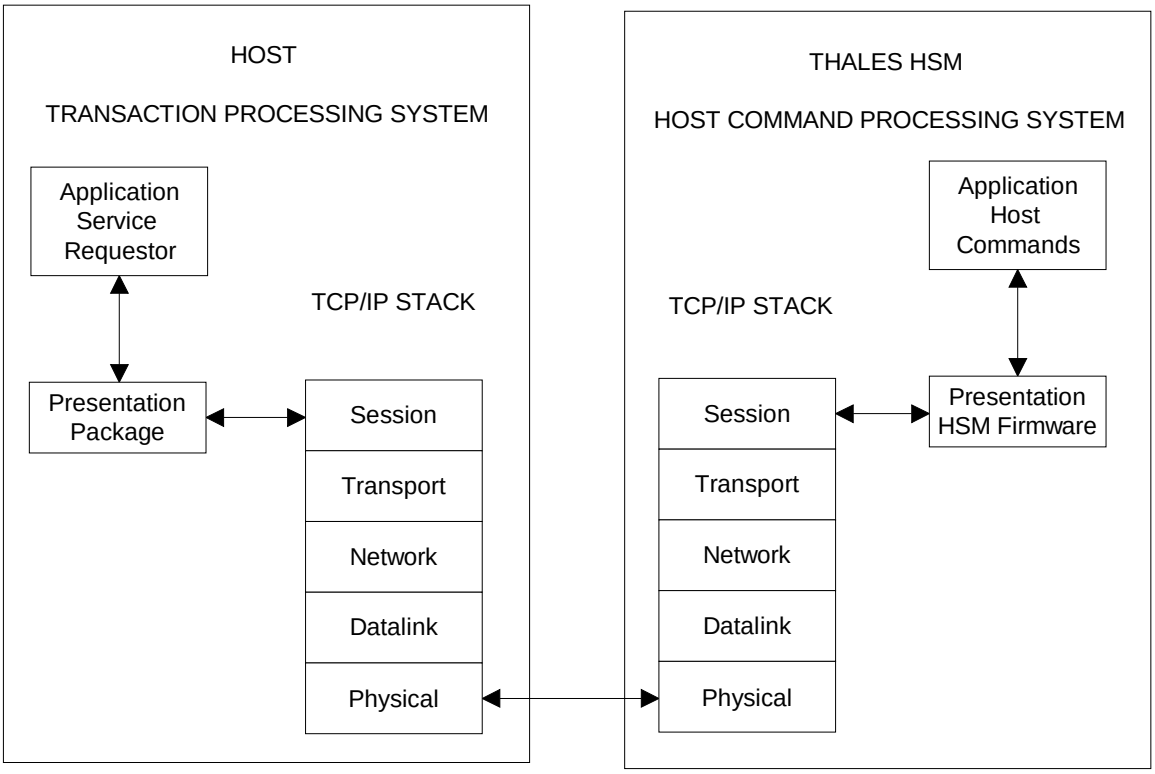
Development and Test Considerations.

The development and testing of HSM based services can be a very costly and time consuming. The problems introduced by the technical complexity of the task, developing processes capable of handling many hundreds of sessions and checking cryptographic output is not normally the easiest of tasks and the delays introduced by having to integrate the Thales firmware design and development group into a project team developing a new security application can easily double the project timescale from say 3 to 6 months with all the corresponding costs.

The way in which system development and testing can be aided, by the use of tools built for purpose, will be briefly examined as a means of reducing timescales and commercial risk within HSM based projects.

Increasing Individual HSM Efficiency

The diagram below shows the path that HSM Host Commands have to follow in order to be executed.



It is obvious from this diagram that both the Host and the HSM have a considerable amount of processing to do in order for a Host Command to be executed on the HSM which gives rise to the greatly reduced maximum transaction rate from 20,000 to 800 for the HSM8-EH HSM.

The options for improving this situation are limited by the fact that all commands have to traverse the IP Stack, and all commands have to also be processed by the presentation layer on each side.

In addition the Host Application, as the service requestor can do little to alter the command throughput .

However, there are two areas which need to be investigated in order to improve the overall systems throughput.

TCP/IP Threads

The number of IP Ports used, and hence the number of threads, can be increased which This will have the effect that the HSM is constantly processing commands.

However, if more that two or three threads are used, this solution will merely increase the amount of transaction queuing within the HSM. Also it is to be noted that Thales have already assumed this approach when quoting their maximum transaction rate of 800 tps.

Therefore, whilst it is a necessary step to take in improving throughput, for a very high performance system, its effect rapidly wears off once more than 4 threads are used. It must also be remembered that the Thales 8000 Series HSM can handle a maximum of 16 TCP/IP Ports and 7000 Series 5 TCP/IP Ports correspondingly.

HSM Host Command Firmware.

This is in effect the inner loop of this service and hence is the core to understanding how the efficiency of an HSM can be improved.

Now, since most of the processing is being performed in the IP Stacks and Presentation Software, the most obvious way of improving transaction processing efficiency is to increase the weighting of the HSM cryptographic processing in the overall equation, in an attempt to reach the theoretical maximum number of transactions per second and the most obvious way of doing this is to increase the number of virtual transactions per HSM Transaction.

Thus, considerable improvements in efficiency can be obtained by blocking the transactions within the HSM Host Command. This however has several effects:

1. The Host Presentation Software becomes more complex, since it will have to track the blocking of the commands prior to execution and despatch on completion.
   The efficiency of this can be considerable increased on a HP NonStop System by running the dispatchers in separate CPUs.

2. The line transmission times will be increased per transaction.
   However this is minimal since the a 100 BaseT link will anyway be able to handle 30,000 tps. using 800 Byte blocks with a blocking factor of 12 transactions per block.

3. The standard command can no longer be used for the HSM host commands to be optimised and therefore there is a requirement for bespoke commands to be implemented to perform the command blocking.
   This in itself has the effect that further efficiencies can be made, for instance not every possible permutation of fields and types are used by the institution wanting to implement the command, hence considerable dead wood can be cut from a command with the corresponding improvement in processing efficiency within the HSM. However, it must be made clear when the firmware is ordered that the firmware corresponding to the command must be implemented as specified and not simply refer to the pre-existing HSM host command. The blocking must also variable, since it is required for system tuning and the maximum efficiency of the HSM firmware.

There is therefore a certain amount of risk associated with these gains however they can be minimised in the following manner:

1. The presentation firmware will be more complex, it will also need to be multithreaded. The best solution to this is to use a pre-existing application framework, preferably written using asynchronous C++. RSI's TELOS product is an ideal candidate for this, has been used in both the financial services and manufacturing industries and has been clocked at performing over 2500 tps on a S-Series NonStop System.

2. The very high level of IP processing means that in extreme cases that HSM network access needs to be split over at least 2 IP stacks and networks, with dynamic presentation layer switching between the stacks for reliability.

3. Optimising a bespoke command can be very expensive and take a considerable amount of time if the standard Thales project approach is taken because:
   a) Each version tried will be treated as a separate project.
   b) Each version will have to be programmed by Thales.
   This can be solved using RSI's HSEMM product whereby a given command can be prototyped in a system and tested before any request to Thales is made to generate new firmware and when the new firmware is generated it can be tested against the prototype.

Maximising the Use of the Available HSMs

The ideal situation for the most efficient use of the available HSMs would have the following properties:

1. All the available HSMs share the given transaction load equally.
   The sum of the number of transactions times their processing time is that same for all HSMs in the System per time slice.

2. The transaction processing time is not affected.
   No given transaction has to wait more than a given amount of time before a response is received. This timeout time will depend on the transaction being processed.

3. The HSM commands processing system is fault tolerant.
   A failure of any given component of the system would not affect the system as a whole to continue to process transactions.

As in all similar engineering situations it is impossible to reach the ideal, the problems which are faced are:

1. The variations throughout the day in host command transaction load cannot be addresses by any fixed configuration method

2. The variation in the host command processing times by the HSM must be considered in any dynamic allocation system to prevent time critical commands being delayed.

3. The maximum despatch rate of about 5000 tps precludes the implementation of the service in a single process.

4. Single points of failure in both host processes and the network infrastructure must be considered to guarantee fault tolerance.

5. The management load of a multiple process system must be considered to prevent it impacting overall performance.

The starting point for the construction of such a system is a knowledge of all the available HSMs their types and their transaction rates.

Using this, a network topology using IP switches can be constructed, so that there are at least 2 networks and the system can continue to operate at peak load even with the loss of one network. Thus, if the HSM set can handle the peak load at 80% busy. There needs to be five networks servicing the HSM set, which generates problems of its own in terms of IP controllers on the Host machines. The solution to this is to buy more HSMs or network infrastructure.

At this point there is a serviceable fault tolerant HSM infrastructure available to the host machines. The next step is to split the HSM transactions into their transaction processing time groups for instance: <0.5 seconds, <1 Second < 5 Seconds, >= 5 Seconds. In order to provide the applications requesting these services to get a timely response to their request.

Using this information then HSMs can be allocated to each group. Note, if this is done statically then the management overhead will be greatly reduced at the cost of some unevenness in the load balancing of the HSMs over time. However, is this is done dynamically then the danger is that every transaction will result in management messages being interchanged. The best possible solution is for this process to be performed dynamically but on a polled basis, say once per minute, with a mixture of statically assigned HSMs, at least one per transaction group, and dynamically assigned for the remainder.

Then the maximum transaction rate to be handled must be determined, say 15,000 tps and the number of CPUs available, say 6. Now, the maximum dispatch rate for a process running on an Itanium system has been estimated at about 5000 tps. Thus at least 15000/5000 = 3 HSM dispatch processes are required. However, this could leave the Host System considerably out of balance. Therefore, in order to balance the Host System it is advised to have a separate HSM dispatch process in each CPU.

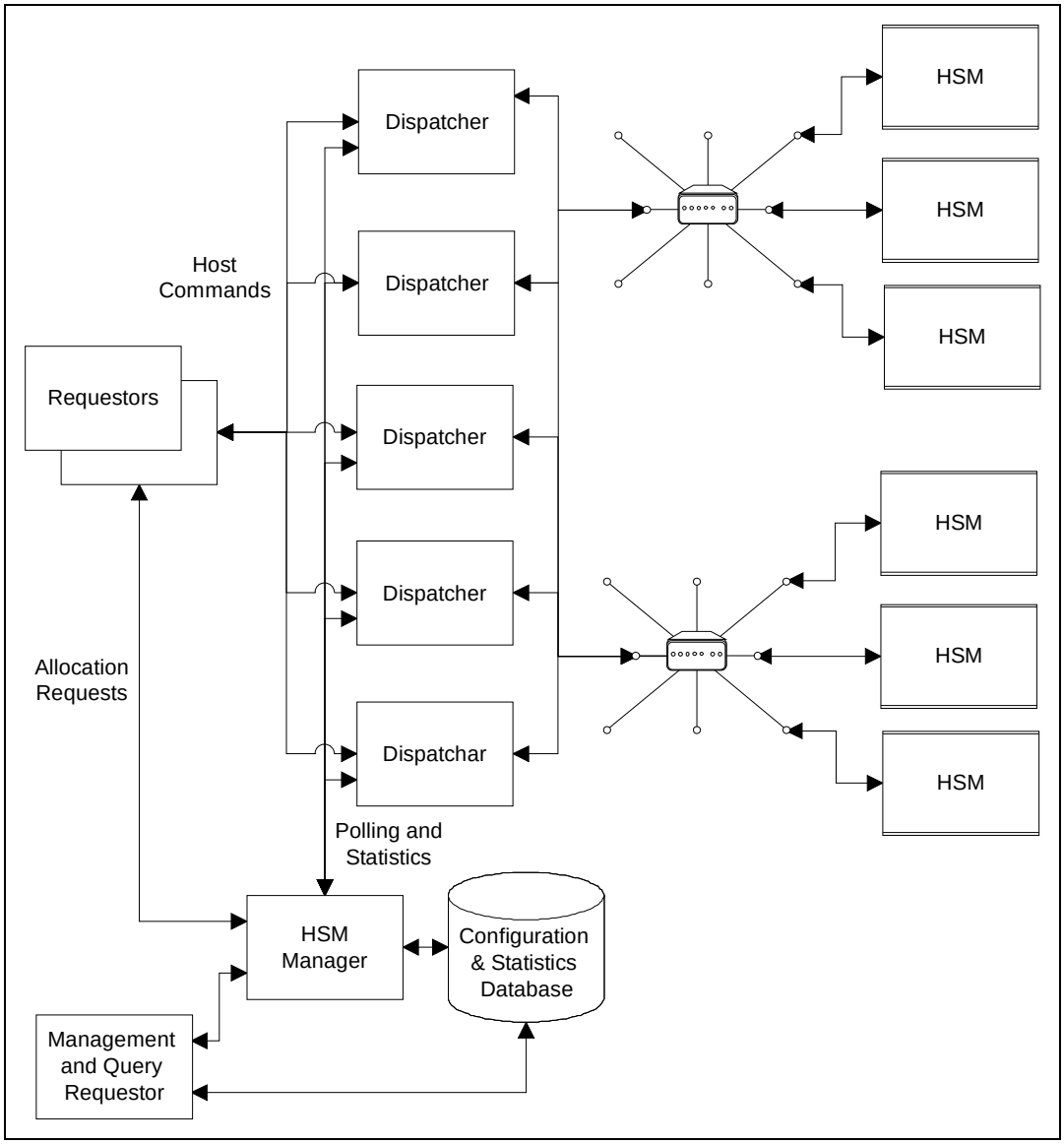
These dispatch processes are best designed so that they have IP Ports open to each of the available HSMs so that all 16 HSM ports for an 8000 Series or 5 ports for a 7000 Series HSM are available to the Host and the order of dispatch is evenly distributed over the available HSMs within each process.

The linkage of application requestors to HSM is done as follows:

Requestors are linked to dispatchers on a round robin basis by means of server requests to the HSM Manager.

Transactions received by the dispatchers are linked by transaction type by the dispatchers. Where the transaction type corresponds to the estimated time for command processing except in the case of blocked commands which are handled as unique transaction types.

The structure of the system is thus:



Management and Reporting

The HSM Manager process performs the overall management of the system and is the point from which information about the operation of the HSM services can be obtained.

System Start-up

The HSM manager process reads the configuration file, if it exists and checks for the presence of the nominated HSM dispatcher processes. It then checks the active processes for their identity (creator, run file…) and starts any missing HSM dispatcher processes. Finally it performs a periodic re-configuration action on the dispatchers using the data from the system start-up file.

Periodic re-Configuration

The configuration of the HSM service is polled at a regular intervals by the manager, once a minute is recommended as an initial value.

As a result of this poll the HSM dispatchers:

1. Return the HSM Port activities of all HSM Ports
   a) Number of Transactions by Command
   b) Transaction Times by Command (Minimum, Maximum, Average)
   c) Port Error information and action taken

2. HSM Ports which are dynamically configurable and have processed no transactions are flagged as down at this point by the dispatcher.

The HSM Manager then determines the new service configuration and sends the individual HSM dispatchers their new configuration for the next time slot. Note. This may allow for the suspension of an HSM to make it available in the next time frame for re-configuration to a new service in periods of high activity.

The configuration and performance data is written to the database in order to make the manager persistent.

Reporting and Control

It is envisioned that the HSM manager will be a Pathway Server to enable performance data to be obtained and the configuration to be inspected and modified, including the addition of new HSMs and dispatchers. Persistence is obtained by the fact that the HSM configuration can be read from the database following a re-start of the HSM manager process by the pathway monitor process.

Application processes will be able to get the names of HSM Service dispatcher processes by using server commands to interrogate the HSM manager. Thereby ensuring an even dynamic distribution of the dispatcher resource at the application level.

HSM Dispatcher Fault Tolerance

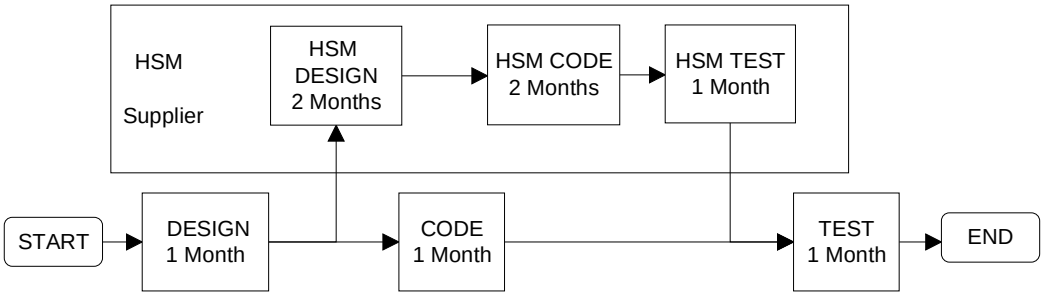
In the case of any dispatcher failing owing to CPU failure the manager re-allocates it to a new CPU and re-starts it as described above. In the case of the CPU becoming available again the corresponding HSM dispatcher process is migrated to the allocated CPU at the soonest possible instance in an ordered fashion to prevent system performance being impacted.

Development and Test Tools

There are major problems in developing high performance applications using bespoke Thales HSM firmware because:

1. It is very difficult to specify and test new cryptographic functions in the HSM
2. The writing of high performance multithreaded applications is not trivial
3. The project run time is greatly increased because of the requirement to manage both the HSM firmware and application components, in which the HSM firmware and design remains on the critical path, with very little opportunity for prototyping and parallel working.

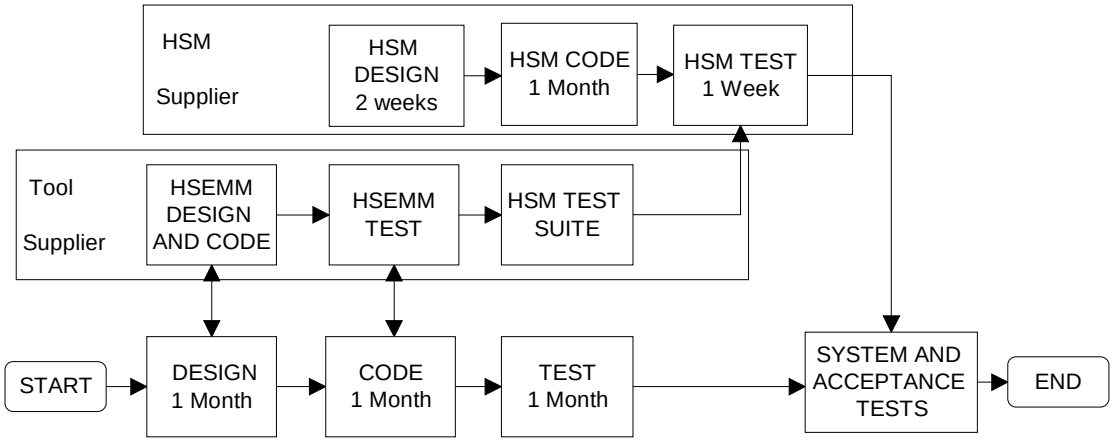
A typical software development project involving HSM firmware development will have the overall project timing as shown below.



This gives a project duration on the critical path of about 7 months in which expensive development personnel are either idle or seconded to other projects.

The principle problem is that HSM firmware suppliers want each stage to be complete before progressing with the next.

However this situation can be altered by using an HSM Emulator and test suite.



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, including the operation of the new HSM Firmware.

A multithreaded framework is also required for the development of the application presentation layer for which a tool is also required.

This need, for the correct tools to prototype the HSM solution and implement the host software so that risk and costs are minimised, is fulfilled by products such as the HSEMM, HSM Test and Development Suite, and TELOS, a C++ multithreaded applications development framework with components, available from Ross Systems International Ltd.

Conclusion

A very high performance, resilient and adaptable HSM Service Architecture can be created, using Thales HSMs linked to HP NonStop systems, which reduces running costs per transaction, maintenance costs, and exposure to reputational and litigation losses.

It has a comprehensive reporting system, which enables managers to determine trends in HSM usage for financial and operational planning purposes.

It is fully compliant with banking rules, regulations and standards such as Basel and PCI DSS in that it is designed to both be secure and avoid service interruptions.

These results can be achieved by:

1. A control and reporting process to manage the HSM Service Architecture.
   It is a fault tolerant process, providing error recovery facilities to the architecture, since it is a Pathway Server and regularly saves the HSM configuration to a database and thus is able to dynamically expand and re-configure the system without interrupting the service.
   It performs load balancing by using dynamic linking techniques between application processes and the HSM Servers.
   It provides both a reporting system and a historic database for analysis of current and historic HSM usage for financial and operational planning.
2. HSM Dispatcher Server processes to manage the linkage between Applications and HSMs attached to TCP/IP Networks.
   It optimises the use of the HSM infrastructure, by using dynamic load balancing techniques based on command load rather than the application stream.
   It also improves throughput by provide a command blocking service from the standard to high performance bespoke HSM commands.
3. TCP/IP Networks to provide high speed delivery of host commands to HSM and fault tolerance in the case of network failure.
4. Bespoke HSM commands to improve the overall throughput of cryptographically light commands, such as PIN verify thereby reducing the overhead of the communications and operating system load and costs per transaction.
5. Using a purpose built application development framework for the dispatchers and HSM Emulation suite to reduce the development costs and increase reliability by thorough testing.

It has the precondition that the application architecture has been correctly designed and configured to allow the high-speed presentation of commands to the HSM Service Architecture in parallel.

It enables any organisation adopting it to tune a Thales HSM service architecture to changing requirements, in volume and crypto mix, both in the short and long term, by a mixture of Dynamic Configuration, to minimises downtime for maintenance and increases performance, Object Technology, which enables the adoption of new crypto technology quickly and safely, NonStop (Tandem) fault tolerant transaction monitoring and message handling services and Cryptographic and Test Tools specifically created to reduced project timescales and improve the quality of the solution provided.

Thus, the proposed system dramatically reduces total cost of ownership of an HSM Service Architecture with a return on investment of considerably less than one year in an adaptable manner to preserve the benefits provided into the future.

