



SECURE EXECUTION ENGINE™
WHITE PAPER



Introduction

Traditional approaches to computer security can result in an uneven spread of security around a company's networks and computers. Parts of the environment may be extremely well protected; others may not, and gaps in security measures can create vulnerabilities. With such a high proportion of mission-critical systems connected to open systems such as the Internet, there are real possibilities that computers can be attacked and misused, data damaged or destroyed, and normal customer service interrupted. Systems, which may be secure when operated on a small scale, can develop weaknesses when they are expanded - or simply lack the scalability that large enterprises require.

Securing computers has a price. All-encompassing systems can lack flexibility, add complexity and time to the development and testing process, and still leave gaps in the security of a system, simply moving the risk to another location.

New types of activity, such as online digital rights management, content distribution, time-stamping, auditing and metering applications, require a new form of logical security where activities at the edge of the network can be secured from the centre, reaching out into the wider world. As mission-critical business activities spread across wide-area networks and are out-sourced to third parties or involve multiple participants, a new, logical approach is needed.

nCipher's Secure Execution Engine™ (SEE) creates a flexible, scalable solution to the problems of creating a large-scale, wide-area secure environment in which only trusted, authenticated application code can operate. SEE expands the security perimeter beyond the key management systems to include the security-sensitive application code. By enabling the creation and secure execution of program code, which can act

as a Trusted Agent™, data can be secured and access monitored and controlled at the edge of the network as well as at the centre. SEE enables the distribution of trust across networks which may not in themselves be secure. Trusted Agents allow you to delegate authority to application processes that you trust, to act on your behalf in server environments that are outside of your direct control.

Based on the exceptional protection provided by the FIPS 140-1 validated nShield hardware security module (HSM), SEE enables developers to create a Trust Appliance™ where keys, server and applications can work together in a secure environment. The security perimeter is redefined to include the trusted code (signed and authenticated) operating together with the HSM to provide an environment where a full range of services can be secured and authenticated.

What is the Secure Execution Engine?

nCipher has developed a new system for protecting the computer code that runs on servers. The Secure Execution Engine (SEE) protects application software as it is executed, and can prevent unauthorised programs from gaining access to a computer or its secure resources. SEE is the first trusted computing platform to provide such a high level of flexibility. In a small-scale PKI or a simple system, it is possible to have trusted human operators oversee critical secure operations, for example the signing of a cross-certification key between two banks. However, trusted human operators cannot oversee every instruction issued by larger-scale systems such as automated trading networks. This creates a situation where the right to use the cryptographic key must be delegated to an application, with no human intervention to check that all is going well.



The fundamental problem, common to many existing HSMs, is that when the keys that they secure are to be used without the explicit intervention of a user, there is no way for the securing HSM to know the source of an instruction. While the HSM provides secure storage for keys, which can lock and unlock data, or certify or verify information, there is no way in most security architectures for the HSM to know what program it is talking to.

How can the HSM know that the application requesting a signature, key or authorisation has permission to do so? The program which runs on the host server may have been compromised or replaced in a manner which the HSM cannot detect. By concentrating security measures solely on protecting the root key stored in the HSM, security gaps may be created elsewhere in the infrastructure.

Ultimately, unless the instructions to the HSM come from a source that is as trustworthy as the HSM itself, and the channel over which those instructions pass is also trustworthy, then the HSM cannot be certain that the source

of these instructions is safe. The SEE sets out to address this problem by closing the security gap between the HSM and applications running on the host server.

Detecting the security gap

This gap between the application running on the host and the HSM is a classic security gap, which could be exploited to attack the system as a whole. Figure 1 shows where the gap between the application code and the HSM occurs in systems without SEE.

As a further example, consider a large-scale public key infrastructure (PKI), with several registration authorities at branch offices communicating with one certification authority at the head office. It is impossible for the security officer to oversee all the transactions in this large-scale system; the certification authority (CA) application code interfaces to the HSM to generate keys and issue certificates without operator intervention.

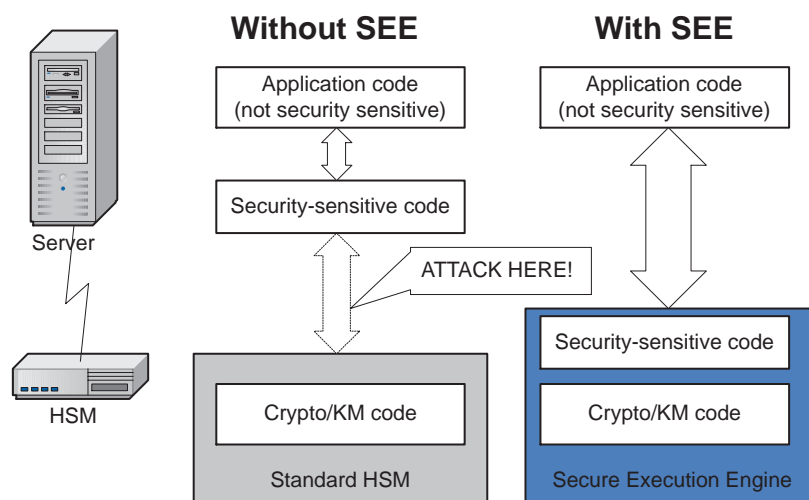


Figure 1 - The SEE expands the security perimeter



Registration authority (RA) servers in the branch offices check the credentials of users and then issue requests for a new certificate to the head office's certification authority server. The head office server issues a new certificate when it receives a signed certificate request from the registration authority servers in the branch offices (Fig.2).

The CA is an unattended process running on a server, to which an HSM is attached. The HSM holds the CA's signing key. The CA application will look at the signed request, and compare it with the security policy management document. If the request received by the certification authority carries a valid signature and is acceptable, according to the policy document, a certificate will be issued. However, if the policy checking is subverted, or the policy document is corrupted, bogus certificates could be issued - although the root key would still be intact in the HSM and there would be no evidence of an attack on the HSM. By placing the policy checking within the

HSM as an SEE application, this type of attack can be avoided.

Existing solutions to the security gap

There are several potential solutions to the security gap between the HSM and applications on the host server, each with potential drawbacks.

One approach is to build a trusted operating system. This typically involves beginning with an existing operating system and building more elaborate security controls onto its existing functionality. There are several problems with this approach, such as obtaining the skilled staff needed to run the specialised system, plus the extra management time required for handling the security needed for system administration tasks. There are also security problems; due to the lack of physical security, trusted operating systems are open to physical attack such as the removal of hard disks.

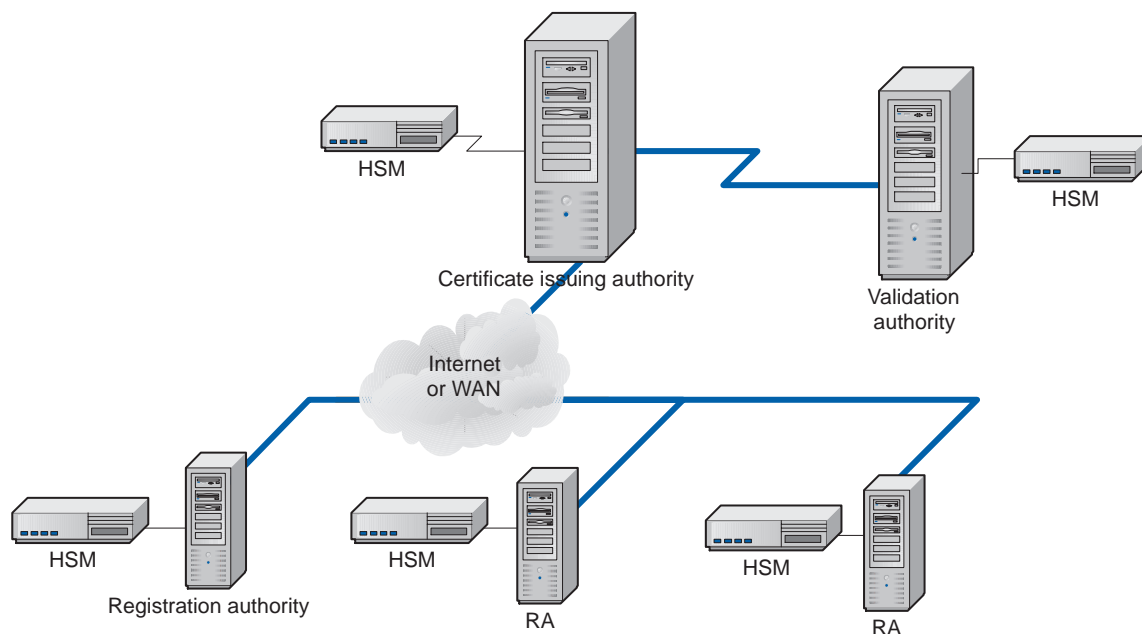


Figure 2 - The SEE in a PKI environment



A second approach is the trusted computing platform. This is a functionally restricted platform with controls on the programs which can be loaded using keys, perhaps checking digital signatures or a certificate chain. In such systems there is typically very little control of the 'trusted' code once it has been accepted and loaded, as it has the free run of the HSM. Another problem for such systems is that the chain of trust often ends with the HSM manufacturer, who holds the root or initialisation key for the module, and could therefore possibly issue certificates acceptable to the HSM.

A significantly more secure solution, employed by SEE, is to move the code inside the same security perimeter that protects the keys, but to restrict the resources which it can access. By tightly controlling what is accessible to the code, for example particular keys, the user is able to define and control the trust model. This control is enforced by digitally signing the code, associating access rights with a specific signing key. SEE allows critical programs to run within the same perimeter, but

because the programs have only been granted limited access, keys are protected from the program as well as from the outside world. Thus the HSM and the server act together to create a Trust Appliance, within which trusted code can operate according to finely tuned policy guidelines.

A thorough system of logical security ensures that a trusted environment is created within which Trusted Agents can operate (Fig.3). Trusted Agents can enforce application-specific security policies on unattended servers, where there is no human means of control to authorise sensitive operations. This also enables the secure automation of secure processes. Trusted Agents are a means of delegating authority to software which you can trust to carry out your orders in a distributed network. By applying digital signatures in the context of Trusted Agents, strong measures of non-repudiation (for example signed audit records) can be established. This is an essential component when providing evidence to third parties for dispute resolution or contractual compliance.

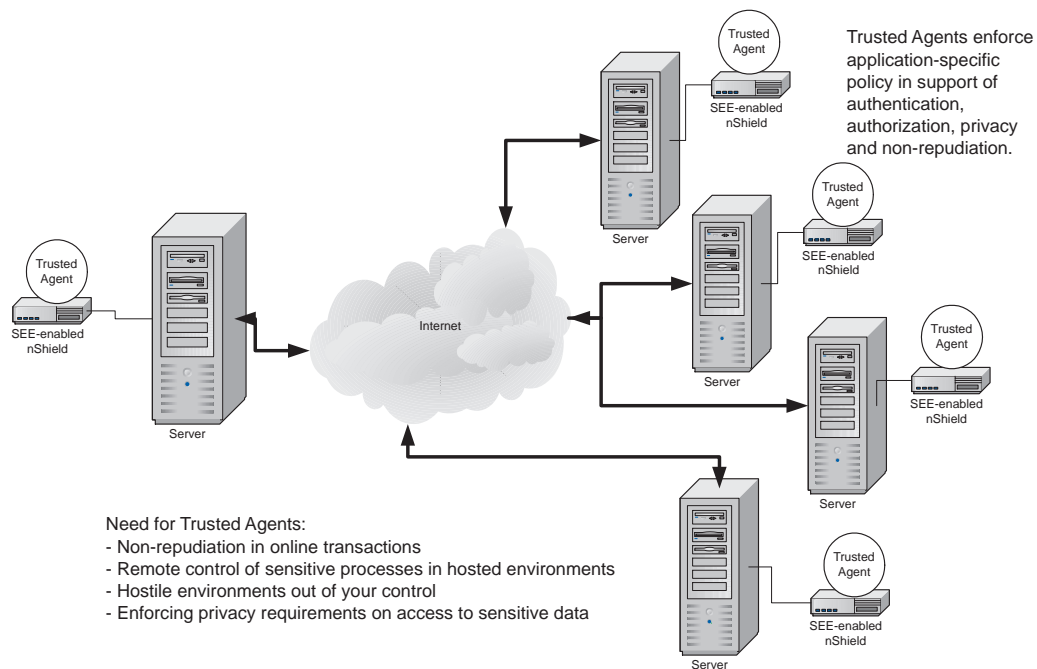


Figure 3 - Trusted Agents in distributed networks



SEE in relation to the nCipher product range and nCipher's Security World

SEE technology is delivered as part of the CodeSafe™ Developer Kit, which includes the tools required for applications to SEE-enable their applications. CodeSafe is part of the nCipher SafeBuilder™ family of developer kits, which enable third-party developers to build custom applications based on nCipher's secure hardware platforms.

SEE technology is built on top of the nCipher secure environment, protected by the nCipher Security World key management environment, a combination of the nShield™ HSM and the KeySafe™ key management software.

The basis for any SEE system is the nShield range of trusted HSMs. The FIPS 140-1 Level 3 versions of nShield are the target platforms for SEE and are labelled 'SEE-ready'. The nShield hardware platform provides a high-performance, scalable computing environment designed specifically for cryptographic applications. Its tamper-evident properties provide a sound physical counterpart to the software and logical security provided by SEE.

KeySafe key management software provides a graphical user interface to the security features of nCipher hardware which enables Security Officers to manage their nCipher Security World more efficiently.

Fundamentals of nCipher key management - the nCipher Security World

Key management within the nCipher hardware security envelope is more flexible and scalable than existing trusted operating systems or computing platforms. In nCipher's Security World, keys are associated with an access control list, which describes their authority based on signatures from other keys. Fragments of the wrapper keys (which protect application keys) can be shared across several smart card tokens, increasing the resilience of the security model (Fig.4). Because there is no pre-installed root key known to nCipher or anyone else, the trust model is better suited to the needs of high security users; it is controlled by the creator of the key, not the creator of the HSM. nCipher's key management uses additional layers of security to protect its own environment and all generated keys.

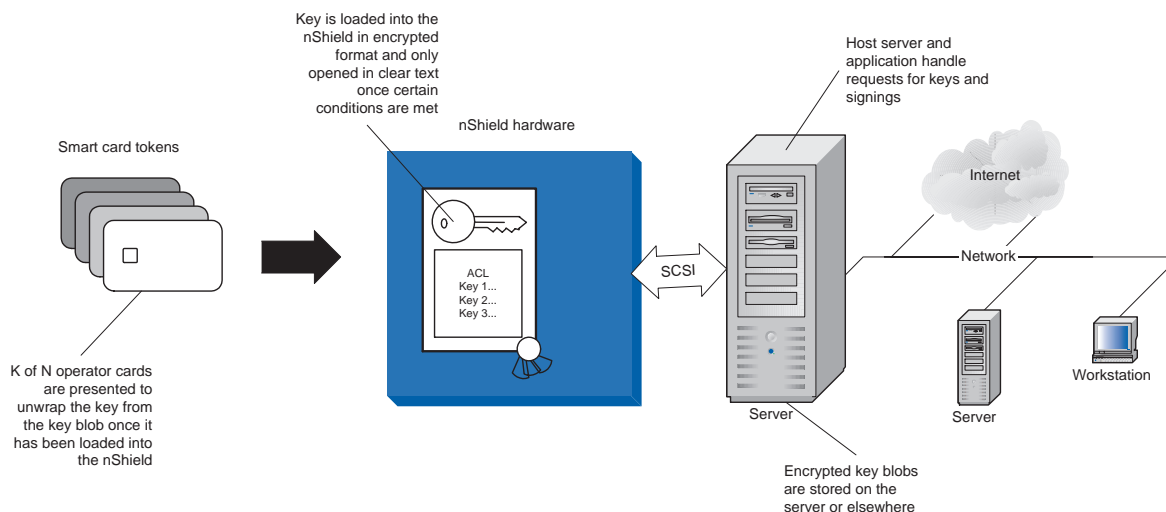


Figure 4 - nCipher Security World Fundamentals



Each key generated by the security module has a corresponding Access Control List (ACL) which is securely bound to it; the two are stored together in an encrypted format known as a 'key blob'. The ACL lists all the actions which the key can be used for, and which operators and keys have the right to initiate those actions. For example, an ACL can specify actions such as signing a message key, and specify that only holders of certain other keys can perform the operation using that key. The ACL is supplied by the creator of the key.

Because access can be controlled at the level of each action with each key, there is exceptionally fine granularity. Ultimately, control over any action rests with whoever created the key; they will have created an access control list specifying the way the key can be used and in what circumstances, and the extent to which the rights enjoyed by the key can be delegated. By specifying who can perform which actions using an access control list, they restrict the right of any program code to performing certain permitted actions.

In some cases this may lead to unattended operation by a program; in others, an action may only be permitted when certain conditions are met, such as the presence of the majority of a set of operator or administrator smart cards (k of n secret-sharing). In all cases, this is determined by the access control list and is ultimately in the control of the party who generated the key.

SEE within the nCipher Security World

SEE extends the concept of access control described above to include application code which is loaded into the module, using a process of delegating rights held in access control lists. The

process of loading the program is not controlled; instead, any piece of code is limited to actions which its signing key allows, i.e. those actions permitted by the access control lists which contain that signing key as an authorised party. If the key which has signed the application does not have permission to do something (via an ACL) the application will not be able to do it either.

Parties in an access control list are identified by their signature keys; if a signature key is used to sign the program code, any rights in any access control lists are delegated to the signed code. In this way, code signed by a key has the right to perform the actions listed in any access control list which contains that key. It is therefore possible to tailor the level or breadth of access granted to any program according to the needs, by using a key with the appropriate level or quantity of access. If the signing key has rights in several ACLs, SEE will permit application code signed by that key the same access to data.

SEE itself sits between the program code and the key resources. Applications never have direct access to the key resources and cannot tamper with them or compromise them. By signing applications and delegating the authority held by the signing key to the application, it can operate as a trusted agent with the permissions and authority granted by the signing key, enabling it to operate securely without live supervision.

SEE can control and secure any program which is required to operate within the security perimeter. This can be either a single nShield HSM or a networked nCipher Security World (where several HSMs are used together with the same key set and security officer to create a larger-scale installation).



SEE real-world examples

Closing protocol gaps: A security gap exists where previously encrypted data is exposed in the clear at the server in order to fulfil an operation or request.

A prime example of a protocol gap is the so-called WAP gap, where the end-to-end security that is taken for granted in SSL is broken at the wireless gateway to allow the wireless security

protocol (WTLS) to be bridged to the SSL protocol. By placing this activity within SEE perimeter, this gap too can be closed with all plain text kept secure within the confines of the nShield HSM, as shown in Figure 5. SEE can be used to close this protocol gap. All decryption takes place inside the nShield HSM, ensuring data does not appear in the clear. Re-encryption also takes place inside the module, delivering end-to-end encryption and protecting data from compromise.

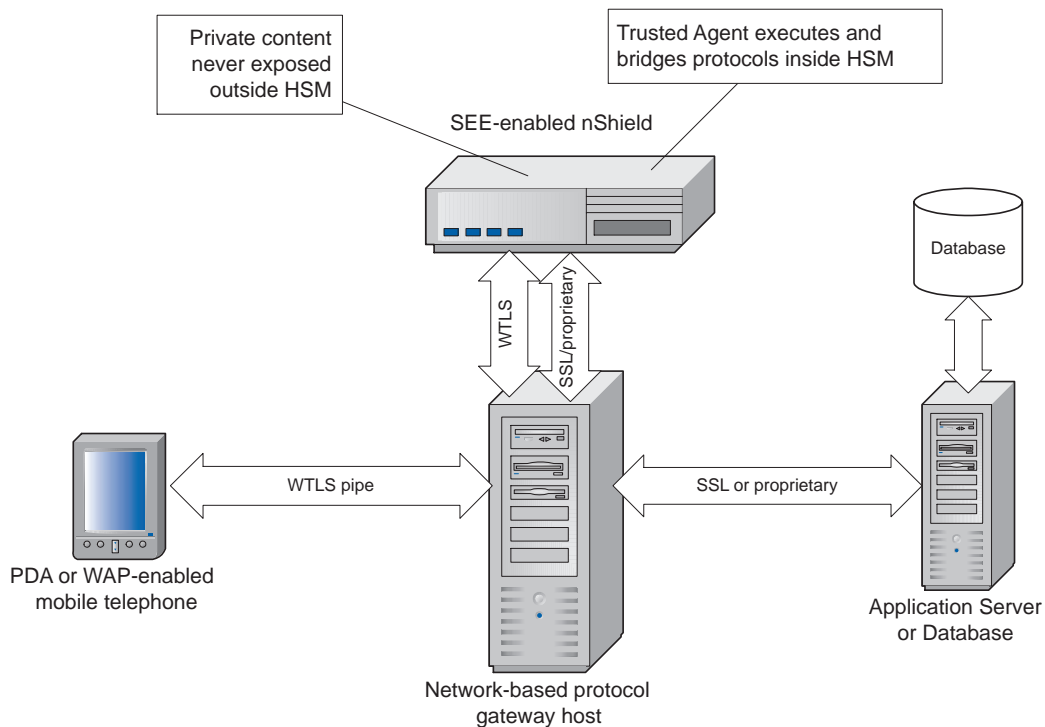


Figure 5 - the SEE and the so-called 'WAP gap'



Consider the example of digital rights management (DRM), which combines the problems of a potential protocol gap and content management (Fig. 6). A subscriber may request a particular piece of digital content, the request being encrypted using SSL. While the details of the requesting subscriber are being compared to the details stored in the (encrypted) database, both are exposed in the clear. Once the subscriber has been authenticated and authorized to receive content, encrypted digital content from a second database must be decrypted, converted to a protocol such as SSL, before being sent to the subscriber. Each of these stages requires a decryption operation at the server, exposing valuable information, before subsequent re-encryption.

Content distribution applications: Consider the example of content distribution applications, where content can only

be delivered to authorised parties, and access to secured content needs to be metered for billing purposes and audited for security purposes (Fig.7). The content and billing servers sit at the heart of the network. At the edge of the network are servers, which handle access to the content and run the Trusted Agents that perform metering and auditing functions. These can be secured by nCipher technology, and then report securely to the billing system server. By using SEE on these servers, only authorised code can perform these sensitive functions, protecting the content and also protecting the integrity of the billing system. There is no scope for tampering with the server software or spoofing it; no spoof can present the appropriate credentials. With a secure audit log there is no scope for authorised users to repudiate any activities; so the system is protected from both authorised and unauthorised users.

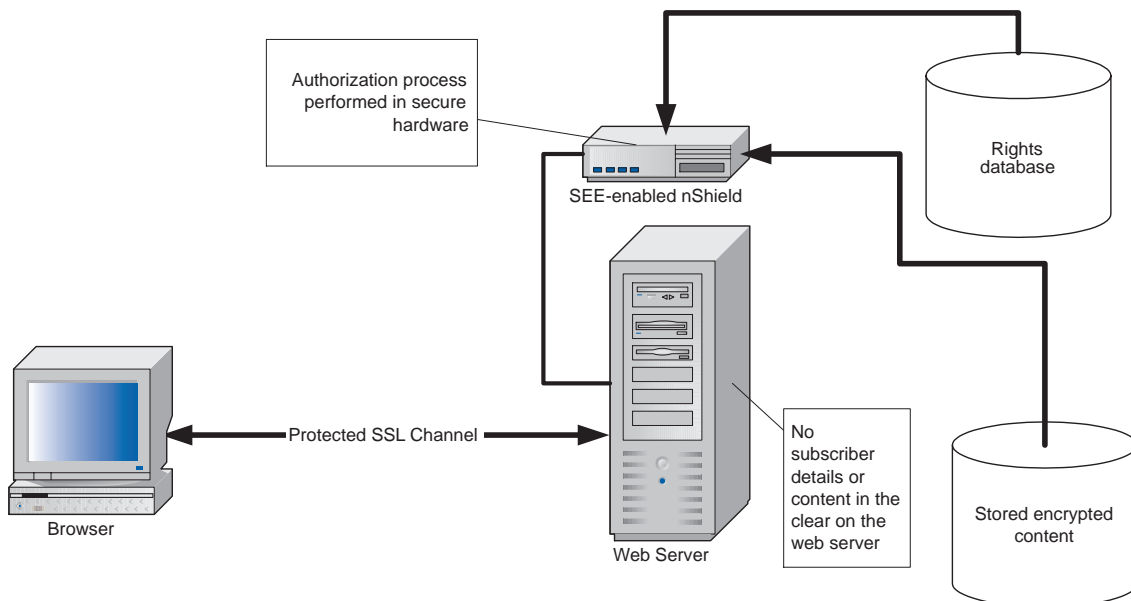


Figure 6 - The protocol gap and digital rights management



Secure audit logs: Audit trails, providing a trusted record of the historical activity of a system and its users, are a vital source of information in many computing applications. Ensuring their security and trustworthiness is critical; if the log can be tampered with, no entry in it can be trusted regardless of whether there has been an actual attack on it. Secure, trusted audit logs are required, which cannot be tampered with. SEE provides a suitable environment for this by using the nShield HSM to sign log entries, providing for authentication and non-repudiation, when using appropriate cryptographic protocols.

Using SEE, whenever an entry is made in the log it can be signed with a timestamp along with a signature on a previous entry. This makes it impossible to tamper with the log, as it becomes impossible to insert or delete entries without breaking the chain of signatures and access rights. Additional features of the HSM hardware, such as secure trusted timestamps, can be used as well, but the fundamental feature is that the audit log is secured by a chain of signatures. Only trusted software providing the appropriate credentials can generate a log entry, ensuring that the log cannot be written to

by users unless they have the appropriate permissions. Typically, the permissions for writing to the audit log would run in parallel with those for using the software being logged.

Combining elements: A simple but effective use for SEE is to enforce business rules, such as actions that must occur simultaneously. This could be used to reduce operator fraud by making it impossible to perform one operation without a counterpart. For example, SEE could be used to bind together two tasks, so that they must be completed together, such as forcing the completion of a debit entry for every credit entry in an accounting system. By enforcing policies on the completion of accounting entries, SEE makes it much harder for rogue operators to subvert the proper operation of an accounting system.

Any attempt to use non-authenticated rogue programs to make false entries in the system would fail, because only signed and authenticated program code would be able to access. In this way, SEE provides two benefits, both that the resources protected by the HSM can only be accessed by authorised code, and that that code can only be operated by authorised users.

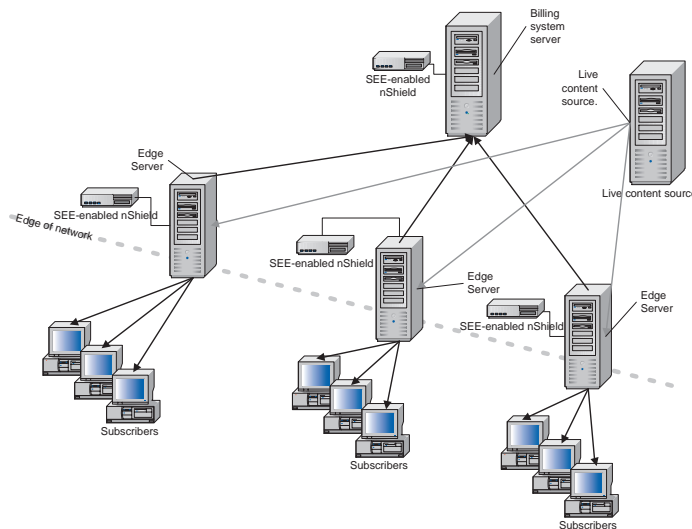


Figure 7 - The SEE in Content Distribution Networks



Enforcing policies in a PKI installation: A typical use for SEE is to secure large-scale, multi-site PKI installations. In the earlier example, it was shown how a registration authority could potentially subvert a certification authority without the CA knowing about it. The CA has a database interface, net connection, a series of policy documents defining the conditions for the issue of certificates, and a signing program as well as their all-important root key. Only the last three are security critical in this model.

To secure such an installation, SEE could be integrated into the CA host server. By providing a verification program for policy documents, the SEE-secured server can read policy documents, then compare incoming requests with them. If a request is acceptable to the policy document, it can be processed, which involves making a signature with the root key, which can only be used by code running in the SEE, signed by the key which authorises its use.

The program code cannot be tampered with without losing rights to access the root key. This approach provides qualitatively better security than previous architectures.

Benefits of SEE for developers

Developing and testing software for trusted environments has always been a complex process. Application code that passes through or operates within a 'black box' environment is difficult to test and debug. The SEE development kit, part of the CodeSafe™ Developer Kit, provides a 'glass box' environment where code can be checked in a 'virtual SEE'. Because the security depends on access control rather than secrecy of operation, it is possible to debug application code without destroying its integrity; it may then be encrypted before roll-out .

nCipher's CodeSafe Developer Kit enables application developers to easily and effectively secure e-business applications in Java™, utilizing nCipher's SEE technology. CodeSafe enables application developers to write Java programs that are securely loaded and executed on an nCipher hardware security module. Programs can be tested and debugged as Java byte code (although it should be noted that the SEE uses its own security features and these are not based on the standard Java security features).

Management facilities enable fine-grained access control and authorization for the use of security critical resources that are protected on the device, such as private keys, non-volatile user memory and hardware-secured time.

With CodeSafe, you can build and deploy Trusted Agents to perform application-specific security functions on your behalf on unattended servers, or in unprotected environments where the operation of the system is outside your direct control. Examples of Trusted Agents include digital meters, authentication agents, time-stamps, audit loggers, digital signature agents and custom encryption processes.

What to do next

nCipher wants to make it easy for you to secure your systems and build trusted computing platforms with SEE. To find out how, contact your local nCipher sales office. Our consultants can help you solve your security problems using our secure systems and also the extensive expertise of our Professional Services Group.

More information about SEE is available on our web site at www.ncipher.com/products.

Abbreviations and glossary

ACL	Access control list
CA	Certification authority
HSM	Hardware security module
Key blob	Encrypted and protected data consisting of a key and its Access Control List, signed by a module
PKI	Public-key infrastructure
RA	Registration authority
SEE	Secure Execution Engine
nCipher Security World	The key management framework within which a set of nCipher HSM hardware and associated software operates
Trusted Agent	Application code operating within the SEE environment which has been signed by a key with authority within the Security World
Trust Appliance	IP network-based appliance where keys, SEE applications and the server are combined to perform a specific security-critical function



CORPORATE HEADQUARTERS

Europe, Middle East & Africa

nCipher Corporation Ltd.
Jupiter House
Station Rd.
Cambridge, CB1 2JD
United Kingdom
Tel: +44 (0) 1223 723600
Fax: +44 (0) 1223 723601
E-mail: sales@ncipher.com

The Americas & Asia Pacific

nCipher, Inc.
500 Unicorn Park Drive
Woburn, MA 01801
United States
Telephone: 1 800 NCIPHER (1 800 624 7437)
or + 1 781 994 4000
Fax: +1 781 994 4001
E-mail: ussales@ncipher.com

Every effort has been made to ensure the information included in this paper is true and correct at the time of going to press. However, the products described herein are subject to continuous development and improvement, and the right is reserved to change their specifications at any time.

©2001 nCipher Corporation Limited. CodeSafe, CipherTools, KeySafe, nCipher, nFast, nForce, nShield and S.E.E. are trademarks or registered trademarks of nCipher Corporation Limited. All other trademarks contained herein are the property of their respective manufacturers.