SECURING ACCESS TO EMBEDDED SYSTEMS
An Effective Concept for Devices Lacking Internet Connection

Bruno Juchli, Peter Sollberger and Roland Portmann
Lucerne University of Applied Sciences & Arts, Technikumstr. 21, 6048 Horw, Switzerland

Keywords: Security, Authentication, Authorisation, Access control, Embedded systems, Certificates.

Abstract: Many embedded systems provide a web interface for maintenance tasks such as system configuration, test execution and firmware updating. Access to this interface usually needs to be restricted to authorized employees. This paper shows an efficient and cost-effective concept to secure maintenance interfaces using widespread standards and technology. By storing authorisation information in standard compliant X.509 certificate extensions *Transport Layer Security* (TLS) and *X.509 Public Key Infrastructure* (PKI) provide mutual authentication, message integrity as well as confidentiality and enable authorisation of employees. Practical experience of the implementation completes this paper.

1 INTRODUCTION

Recent years have shown a drastic rise of embedded systems which are found in elevators, network routers, automotives, equipment of building automation and many more. Most of these systems require an interface to access logs, status and configuration information and to enable replacement of firmware. An exemplar infrastructure of an organization might look like shown in Figure 1.

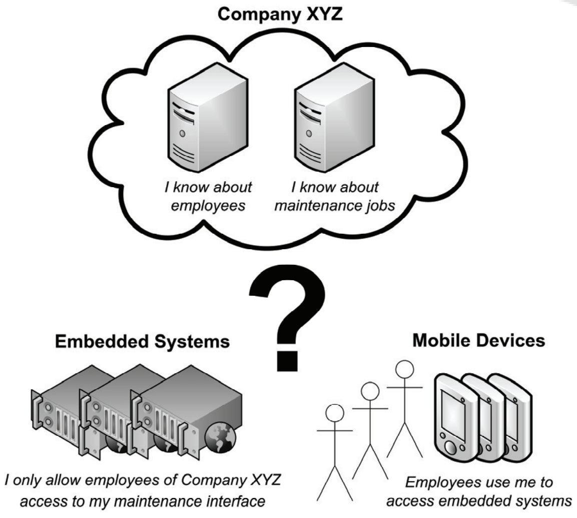

Company XYZ
I know about employees
I know about maintenance jobs
?
Embedded Systems
Mobile Devices
I only allow employees of Company XYZ access to my maintenance interface
Employees use me to access embedded systems

Figure 1: Exemplar infrastructure of an organisation.

Company XYZ services multiple devices. The company has databases that store information about employees and maintenance jobs. Employees use mobile devices such as laptops, smartphones or PDAs to access the maintenance interface of the embedded system on site.

To protect maintenance interfaces it is common to use simple techniques like password authentication or the usage of proprietary protocols (often referred to as *security through obscurity*). These methods mostly do not withhold a thorough security check and/or pose substantial disadvantages like password management overhead and risk of password theft.

The hereafter presented concept considers technological as well as business process aspects. These concrete goals will be satisfied:

- Authenticate employees on the maintenance interface
- Enable precise authorisation for features of the maintenance interface
- Ensure integrity of transferred firmware and configuration (files)
- Require a minimum of management overhead
- The protected device does not require an internet connection

Much like in (Hsu, 1997) short-lived certificates are used to reduce management cost and carry authorisation information, instead of only providing for authentication. However, the hereafter presented

Juchli B., Sollberger P. and Portmann R.
SECURING ACCESS TO EMBEDDED SYSTEMS - An Effective Concept for Devices Lacking Internet Connection.
DOI: 10.5220/0002779401520158
In *Proceedings of the 6th International Conference on Web Information Systems and Technology (WEBIST 2010)*, page
ISBN: 978-989-674-025-2

concept applies a different certificate issuing process and concentrates on securing access to offline and off-site embedded systems, practical implementation and experience and also offers a secure approach for outsourced maintenance.

2 TECHNOLOGY OVERVIEW

In this section a basic overview of the technologies/standards used is given: Public key cryptography, Transport Layer Security (TLS) and X.509 Public Key Infrastructure (PKI).

2.1 Public Key Cryptography

Public key cryptography employs an asymmetric key pair for encryption and decryption. The key pair consists of a private and a public key. When a message was encrypted with one of the keys, the decryption can only be done using the other key. As the name states, the private key is kept private by the owner while the public key is publically available. Please note that while the keys are mathematically related the effort to calculate one key from the other is too big to be practically feasible.

Confidentiality. A Person A (Alice) can send a confidential message to Person B (Bob) by encrypting the message with Bob's public key so the message can only be decrypted using Bob's private Key.

Authenticity. If we want proof that the message really came from Alice, Alice can create a hash of the message and encrypt it using her private key. Bob will create a hash of the message himself. He will decrypt the hash using Alice's public key. If both hashes are the same the message is indeed from Alice and was not tampered with.

These methods can of course be combined to achieve both confidentiality and authenticity (Choudhury, 2002).

2.2 X.509 Public Key Infrastructure

The X.509 Standard is an extensive specification of a PKI. An overview on version three of the standard is given hereafter. For further details please see (Cooper, 2008).

A **certificate** basically is a temporary valid association of a public key with identity Information (such as name, address etc.). Besides the public-key and subject information a certificate also contains: issuer and his signature, serial number, valid from, valid until and optionally extensions.

A **certificate authority** (CA) is an entity that is issuing certificates to subjects after checking their identity. The certificate authority vouches for the validity of the certificates issued. Each certificate authority also owns a public private key pair (and an associated certificate) which is needed in order to create certificates.

Certificate theft occurs when the private key of a certificate is compromised and enables an attacker to impersonate the subject of the certificate.

2.2.1 Certificate Extensions

X.509 allows adding custom extensions (called private extensions) to certificates. These extensions are protected from falsification – either from the owner of the certificate or a third party attacker – since they are incorporated in the calculation of the digital signature. This makes extensions predestined to carry authorisation information. It can also be used to store protected configuration values and even new firmware images that need to be uploaded to an embedded system. Private extensions can be put in the *extensions* field of the certificate. An extension can only occur once in a certificate and basically is an ASN.1 structure. See (Objective Systems, 2003) for an overview of ASN.1 structures.

An extension can be marked as either critical or non-critical. According to RFC5280 (Cooper, 2008) a system encountering a critical extension it doesn't know must reject the certificate.

Please note that both, the amount of extensions and the maximum length of an extension are not specified by a standard but rather are implementation dependent.

2.2.2 Certificate Chains, Certificate Verification

Technically every certificate can be used to create (issue) multiple new certificates in turn. Thus, certificates can be organised in a tree hierarchy. To check the validity of a leaf-certificate all certificates on the path from the leaf to the root certificate need to be checked for validity. These certificates constitute a **certificate chain**.

In an X.509 PKI only certificate authorities may create certificates. This means that all certificates except the leaf certificate must be CA certificates. This is identified by the *basic constraints* extension (Cooper, 2008).

The CA at the top of a tree structure is called a **root CA**.

Only certificates that were issued by a **trusted CA** or one of its subordinate CAs are accepted. Thus, all certificates belonging to a branch of a trusted CA are accepted. A company can define which CAs they want to trust.

Figure 2 shows an exemplar tree hierarchy and the certificate chain (marked grey) for *Leaf 1-1-1*.

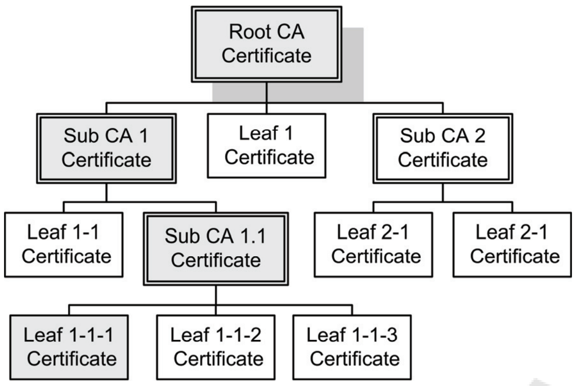


Figure 2: Certificate tree hierarchy.

2.3 Transport Layer Security

Transport Layer Security (TLS) provides endpoint authentication and communication confidentiality for network communications by employing public key cryptography and X.509 certificates. One of the most common applications of TLS is securing connections between web-servers and browsers by HyperText Transfer Protocol Secure (HTTPS). In a typical online banking scenario, when a client establishes an HTTPS connection to the bank's server, the server will present a certificate to prove his identity. If the certificate is invalid, expired, was not issued by a trusted Certificate Authority or was not issued for this domain the connection will be closed by the client.

TLS also offers **mutual authentication**, where both, the client and the server need to authenticate by certificates. Mutual authentication is not very common – but will be shown to be very useful in this paper.

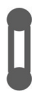
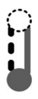
For more details on HTTPS and TLS please see (Rescorla, 2000) and (Dierks, 2006) respectively.

3 AUTHORISATION SYSTEM

First, a simple approach is shown. This simple approach possesses some flaws which are documented. Then, a second approach that was evolved from the first is presented. The evolved approach will eliminate most of the first approach's flaws. This should simplify understanding the complete concept and will account for design decisions. The third subsection will show an approach to grant restricted access to third parties. The fourth subsection will discuss general security and economic considerations.

In the drawings the connectors shown in Table 1 are used to differentiate between TLS connections with and without mutual authentication:

Table 1: TLS connection drawings.

	This is a mutually authenticated connection. Both endpoints need to present a valid certificate
	Only the server is authenticated by a certificate. The server's end is represented by the filled circle. The client authenticates with the web interface by username and password

3.1 Simple Approach

This approach is technically quite straight forward: There is an authorisation system, which knows all protected embedded systems, employees and maintenance jobs. It is issuing extended client certificates to employees. A private extension (non-critical) is added to the client certificate. The extension contains the employee's authorisations to protected devices for the next 14 days. When an employee connects to the HTTPS/TLS secured web interface of a protected device, the protected device asks the employee to present a certificate issued by the root CA. The protected device authenticates the employee and searches the certificate's extension for an authorisation entry matching the protected device's identifier. If one is found, the employee is allowed access to the interface.

Please note, that the protected device needs a server certificate since the TLS server always needs to authenticate. The protected device also has the root CA certificate marked as a trusted certificate.

Technically, this system works quite well. There are two security concerns:

- When a certificate is stolen from a mobile device, the thief may access (some) protected devices until the certificate expires (14 days).
- When the root CA certificate is stolen the thief may access all protected devices until all certificates that were issued by the root CA are replaced. This includes the certificates on the protected devices.

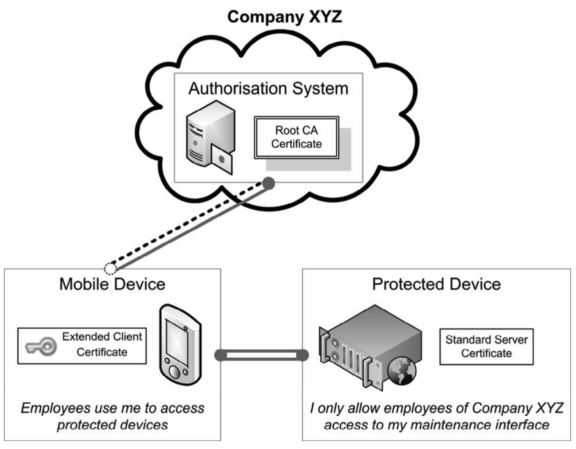

Company XYZ
Authorisation System
Root CA Certificate
Mobile Device
Extended Client Certificate
Employees use me to access protected devices
Protected Device
Standard Server Certificate
I only allow employees of Company XYZ access to my maintenance interface

Figure 3: Simple approach.

However, besides the security concerns this system also has its management caveats. A process to install the client certificates on the employee's mobile devices is needed. The process requires the presence of the employee, his mobile device and an administrator who has the duty to check the identity of the employee. The administrator will open a browser window on the employee's mobile device, connect to the authorisation system's web interface and logon by using a username/password combination. The administrator then installs the employee's extended client certificate and hands the device over to the employee. Since the certificate only contains the authorisations for the next 14 days, this process will have to be repeated every 14 days (for every employee). If the authorisations (maintenance jobs) change during such a 14 day period, the process needs to be repeated as well. This results in two more concerns:

- Excessive management overhead makes this approach uneconomical
- An attacker might place a key logger on the mobile device since the administrator will enter his password regularly every 14 days. If the password was chosen unwisely, this could lead to the theft of the root CA's certificate and potentially pose risk to the entire infrastructure.

Of course there is the possibility to just add authorisations for a whole year – but this would prevent the use of fine granular permissions (like "person B is authorised to access the logs of device D from 01.05.2009 06:00 until 01.05.2009 24:00") and increase the damage in case of client certificate theft.

3.2 Evolved Approach

Most of the former approach's problems are connected to the validity period of the client certificate: If set too low, it will result in lots of management overhead. If set too high, security is compromised. To circumvent this issue, two kinds of client certificates are used:

- Authorisation certificate: extended client certificate that is valid for one day only. As previously, the protected device will request this certificate for authentication and authorisation.
- Identity certificate: standard client certificate that is valid for five years. This is used to authenticate the employee at the authorisation systems web interface.

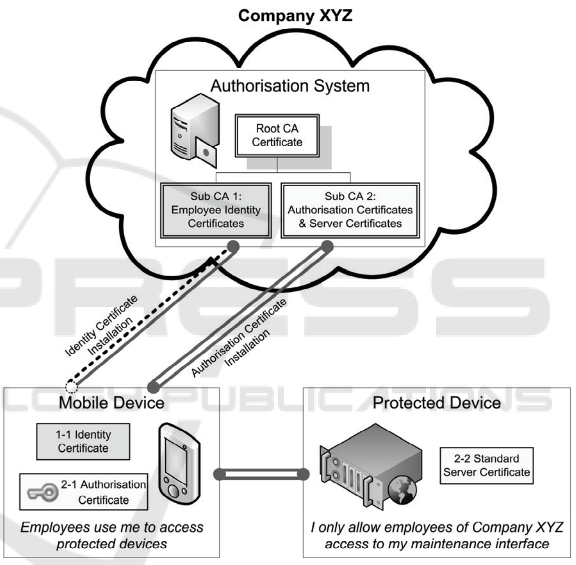

Company XYZ
Authorisation System
Root CA Certificate
Sub CA 1: Employee Identity Certificates
Sub CA 2: Authorisation Certificates & Server Certificates
Identity Certificate Installation
Authorisation Certificate Installation
Mobile Device
1-1 Identity Certificate
2-1 Authorisation Certificate
Employees use me to access protected devices
Protected Device
2-2 Standard Server Certificate
I only allow employees of Company XYZ access to my maintenance interface

Figure 4: Evolved approach.

The former certificate deployment process is now used to install the identity certificate on the employee's mobile device - once every five years only.

Another process is needed to update the authorisation certificates, since they are valid for one day only. This process can, and should, be fully automated. The process is as follows: every day when the mobile device is powered-up for the first time, it connects to the authorisation system using a mutually authenticated channel. The mobile device asks for a new, up to date, authorisation certificate. The authorisation system will grant the request if the client presents a valid identity certificate. The authorisation system uses the information in the identity certificate to find out which jobs are assigned to the employee and thus which authorisations he needs. These authorisations will be included in the certificate and the certificate will be sent to the mobile device. The mobile device will then automatically install the new authorisation certificate – the employee is ready to access protected devices.

One more process is needed in case of client certificate theft: If an employee's certificate is stolen the admin will revoke the concerned certificate on the authorisation system. The admin will use the already introduced process to install a new identity certificate on the employee's mobile device.

As you can see in Figure 4 two sub certificate authorities were added to the Authorisation System. This is because of the TLS protocol: When the server requests a certificate from the employee, the server tells the client which CAs he trusts. If he would specify that he trusts the root CA, the client would indeterminably send either an identity- or an authorisation-certificate to the server and authentication might fail. The two sub-CAs enable the server to specify which kind of certificate he needs.

As you can see this approach is much more efficient and secure:

- Management overhead was drastically reduced to a minimum.
- The thief of a client certificate can exploit the authorisations of the certificate he stole during one day only.

One security issue remains: The theft of CA certificates. This event is less likely but much more drastic than the theft of a client certificate. Please see the next section for improvement possibilities.

3.3 Authorising Third Parties

In many cases granting access to third parties is desired (e.g. maintenance outsourcing) or even a statutory requirement. In these cases use of certificate hierarchies can be made: the company's CA can issue a third party sub CA certificates. The third party can then issue certificates to their own employees. If restrictions (e.g. grant only access to particular models, do not allow firmware replacement etc.) should be imposed to the third party, the super CA can place these restrictions into an extension of the sub CA certificates. Since the complete certificate chain needs to be presented to the device, the device can check every certificate for restrictions. Figure 5 shows the certificate verification and authorisation extraction process.

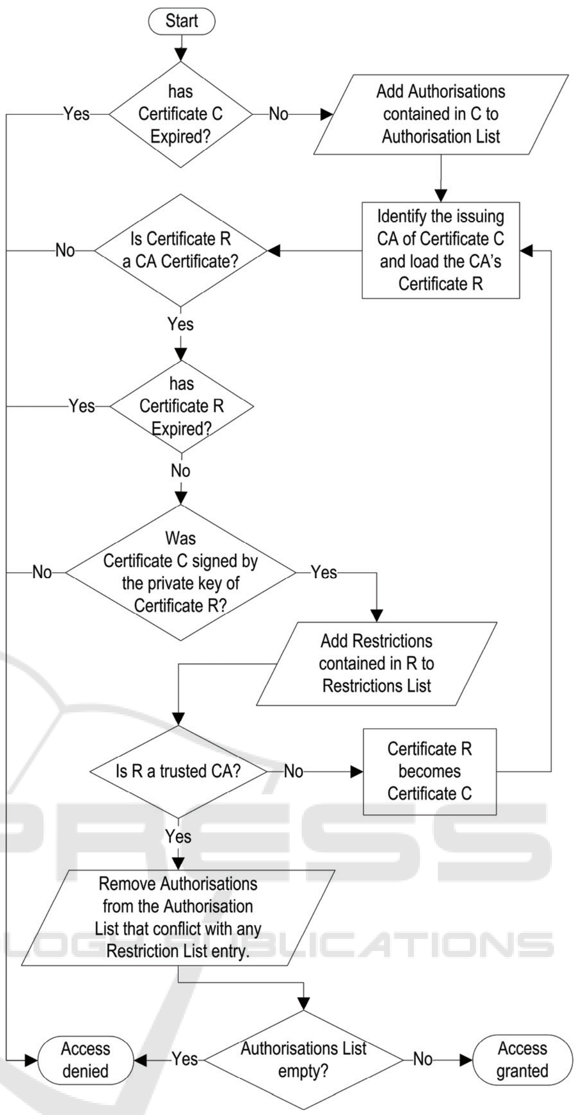

Start
has Certificate C Expired?
Yes
No
Add Authorisations contained in C to Authorisation List
Identify the issuing CA of Certificate C and load the CA's Certificate R
Is Certificate R a CA Certificate?
No
Yes
has Certificate R Expired?
Yes
No
Was Certificate C signed by the private key of Certificate R?
No
Yes
Add Restrictions contained in R to Restrictions List
Is R a trusted CA?
No
Certificate R becomes Certificate C
Yes
Remove Authorisations from the Authorisation List that conflict with any Restriction List entry.
Authorisations List empty?
Yes
Access denied
No
Access granted

Figure 5: Certificate Verification and Authorisation Extraction.

3.4 Security and Economic Considerations

3.4.1 Certificate Validity Period

The Certificate validity period is an important criterion in security as well as in management costs. Sooner or later all of the certificates (including root-CA, sub-CA's and server certificates) need to be replaced. This is due to the fact, that public-key algorithms with a given key-size become less secure with the increasing availability of computation power (thus dependent on time). We need certificates to expire before an attacker can compute the private key of the certificate. Thus it is recommended to use very strong keys and encryption for CA certificates.

When a CA certificate expires, the CA certificate and all its subordinate certificates need to be replaced. This can result in massive costs since every protected device has to be visited to replace its certificates.

In conclusion, the **CA certificate** should be set to be valid until somewhat before the private key becomes unsafe. A validity period of 20 to 30 years is common for root CA certificates (VeriSign, 2009) (GeoTrust, 2009).

Validity period of the **protected device's server certificate** is not a security concern and can thus be set to expire at the expiration date of its parent certificate.

Regarding the validity period of the **employee's authorisation certificate** the following influences need to be considered: damage of certificate theft, availability of internet access from the mobile device, frequency of changes in schedules and granularity of authorisations. If very specific authorisations are employed the damage of certificate theft is smaller and a longer validity period can be used. Given today's high availability of mobile internet access a validity period between one and three days is feasible and economically bearable.

Please note that a certificate is never valid longer than the certificate of its certificate chain expiring the earliest.

3.4.2 Certificate Theft Protection

CA Certificates. The certificates need to be put in an encrypted storage. The CA certificates should use a strong public-key algorithm with large key-size. Access to the authorisation system (CA) should be restricted – make it accessible only from intranet (use VPN for access from outside). Log all access to help uncovering the theft of a client certificate.

Client Certificates. The employees mobile devices should be protected by password login and the certificates and private keys should be stored encrypted. When a new identity certificate is issued for an employee, all of his old identity certificates should be revoked. If a new identity certificate is installed because the old was stolen (and revoked!), the employee's mobile device should be reset to make sure it doesn't contain any malware like key loggers or trojans. Otherwise the administrator might logon to the authentication system using an infected system which would enable the attacker to steal the administrator's credentials and compromise the system.

Server Certificates. No special protection is required.

4 EXPERIENCE

The DustBot project (http://www.dustbot.org) is aimed at designing, developing, testing and demonstrating a system for improving the management of urban hygiene based on a network of autonomous and cooperating robots, embedded in an Ambient Intelligence infrastructure. These robots will be able to clean streets and collect small quantities of home garbage from citizens. The concept shown in this paper was successfully applied to secure the access to the Linux (Ubuntu Server) based DustBot robots. It was required that the robots could be accessed even (and especially) if their network links were down, for example to diagnose the underlying problem of a network link failure. As mobile devices ASUS EeePC 1000HE netbooks were chosen for their price, long battery runtime and wireless connectivity. Equipped with a WAN interface, the mobile device can request a new authorisation certificate at any time. Bluetooth Personal Area Network (PAN) was employed to connect the maintenance interface to the robots. Each robot acts as a PAN Network Access Point (NAP), similarly to a WLAN Access Point. On Ubuntu, setup for a PAN NAP is just a matter of installing precompiled packets (bluez, www.bluez.org) and configuration – and was thus an easy task. The robot's maintenance web interface is served by lighttpd, a lightweight HTTP Server (www.lighttpd.net). The web interface was implemented in Python. The certificate authorisation extraction is done using the Python ASN.1 library pyasn1 (http://pyasn1.sourceforge.net/).

We experienced two (non critical) usability constraints in the implementation:

- HTTPS does not allow displaying a custom error page when a connection fails due to an unaccepted or invalid certificate. In such an event the connection is terminated and the browser displays an error.
- Firefox 2-3 and Internet Explorer 6-8 (others were not tested) do not offer an automated mechanism to remove expired certificates from their certificate stores. If authorisation certificates are issued frequently the certificate

store gets cluttered. The same issue was noted by (Hsu, 1997).

Firefox was configured to automatically select the client certificate to present to the server. This works flawlessly as Firefox always chooses the latest certificate. It also enhances usability since the user no more needs to choose a certificate every time when connecting to the maintenance interface.

No problems concerning the functionality were found and the implementation was proved to work justly. Furthermore, no compatibility issues arose from the use of private extensions marked as non critical. Also, both crypto libraries utilised in the project, openssl (http://www.openssl.org) and Bouncy Castle crypto API (http://www.bouncycastle.org), proved to be capable of handling an extension the size of 100 mebibytes ($100 \cdot 2^{20}$ bytes), which should in most cases suffice for a massive amount of authorisations, configuration values or even a firmware image.

5 CONCLUSIONS AND FUTURE WORK

The concept shown in this paper is configurable, adjustable to an organisation hierarchy, scalable and allows very fine granular authorisations. Since most of the concept's processes can be automated the process can be implemented to be very cost efficient. Also, practical experience shows that the concept can easily be implemented using today's available hardware and software infrastructure.

However, there are some caveats that have not been addressed by the concept. These are mainly:

- Detection of certificate theft
- An efficient process to replace the certificates in case of CA certificate theft or CA certificate expiration. An automated process could be implemented if the protected devices possess internet access.

These caveats need to be addressed in order to achieve a system that can reliably and efficiently be used throughout decades.

ACKNOWLEDGEMENTS

This work was supported in part by the EU STREP FP6-IST-045299-STREP "DustBot".

REFERENCES

Choudhury, S. 2002. *Public Key Infrastructure: Implementation and Design*. Wiley.

Cooper, D. et al. 2008. *Internet X.509 Public Key Infrastructure: Certificate and Certificate Revocation List (CRL) Profile*. [Internet] Available from: <http://tools.ietf.org/html/rfc5280> [Accessed 21 October 2009]

Dierks, T., Rescorla, E. 2006. *The Transport Layer Security (TLS) Protocol Version 1.1*. [Internet] Available from: <http://tools.ietf.org/rfcmarkup?rfc=4346> [Accessed 21 October 2009]

Hsu, Y.-K. Seymour, S. 1997. Intranet Security Framework Based on Short-lived Certificates, in *Proceedings of the Sixth IEEE Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society, pp. 228-233.

GeoTrust. 2009. *Root Certificates*. [Internet] Available from: <http://www.geotrust.com/resources/root-certificates/index.html> [Accessed 21 October 2009]

Objective Systems. 2003. *ASN.1 Tutorial*. [Internet] Available from: <http://www.obj-sys.com/asn1tutorial/asn1only.html> [Accessed 21 October 2009]

Rescorla, E. 2000. *HTTP over TLS*. [Internet] Available from: <http://tools.ietf.org/html/rfc2818> [Accessed 21 October 2009]

VeriSign. 2009. *Root Certificates*. [Internet] Available from: <https://www.verisign.com/support/roots.html> [Accessed 21 October 2009]