Public Key Infrastructure Issues for Enterprise Level Security

Kevin Foltz and William R. Simpson
Institute for Defense Analyses, 4850 Mark Center Drive, Alexandria, VA 22311, U.S.A.

Keywords: Enterprise, Public Key Infrastructure, System Design, Application Security, Security, Distinguished Name, X.509, Certificate, Zero Trust Architecture.

Abstract: A public key infrastructure (PKI) provides a way to manage identities within an enterprise. Users are provided public/private key pairs, and trusted certification authorities issue credentials binding a user name to the associated public key for that user. This enables security functions by users within the enterprise, such as authentication, signature creation and validation, encryption, and decryption. However, the enterprise often interacts with partner enterprises and the open web, which may use different PKIs. Mobile devices do not easily operate with hardware-based PKI tokens such as smartcards. Standard digital signatures lack timing information such as validity or expiration. This paper examines some of the security challenges related to PKI deployment in the context of Enterprise Level Security (ELS), an enterprise solution for a high security environment.

1 INTRODUCTION

Public key infrastructure (PKI) is a commonly used method for managing entity identities in large enterprises. Standard PKI components include:

- Public key certificates
- Certificate repository
- Certificate revocation
- Key backup and recovery
- Non-repudiation of digital signatures
- Automatic update of key pairs and certificates
- Management of key histories
- Support for cross-certification
- Software implementation to use items listed above

Together, these form the basis for an automatic, transparent, and usable PKI (Entrust Datacard, 2019) (Cooper et al., 2018).

The use of PKI is widespread on the public web, as well as within enterprises. Web servers use certificates from trusted certification authorities (CAs) to authenticate to users connected from remote locations through potentially untrusted or hostile networks. Enterprises use PKI to provide employees, servers, services, and other entities with a convenient way to encrypt and decrypt data, sign and verify content, and perform third-party authentication, where neither party has an established relationship prior to the authentication.

Enterprise Level Security (ELS) is an approach for enterprise security that builds from basic principles and concepts. PKI is one of the key building blocks for ELS. However, for high security, many issues arise in a practical PKI implementation that are not commonly addressed. Naming of entities in an enterprise is a core function that PKI relies on. This affects not only PKI functions within an enterprise, but interactions with partner organizations and the web in general. Credential management with mobile devices opens new opportunities and challenges. Common functions, such as signatures, in raw form lack many desirable properties related to time and transitioning roles and responsibilities.

This paper examines some of the key issues for ELS PKI implementation. It identifies issues with certain enterprise situations and describes solutions to maintain security properties within the ELS-enabled enterprise. This is part of a larger effort to secure information sharing for the United States Air Force (Foltz and Simpson, 2017; Foltz and Simpson, 2016a, Foltz and Simpson, 2016b).

Foltz, K. and Simpson, W.
Public Key Infrastructure Issues for Enterprise Level Security.
DOI: 10.5220/0009342000910098
In *Proceedings of the 22nd International Conference on Enterprise Information Systems (ICEIS 2020) - Volume 1*, pages 91-98
ISBN: 978-989-758-423-7

2 ELS BACKGROUND

ELS is an architectural approach for designing security into a distributed web-based enterprise system of information sharing. It focuses on identity and access management as the root of security. PKI is an important part of ELS, because it is the preferred method for managing identities and performing authentication. However, ELS is much bigger than PKI. The key elements of ELS are identified in this section (Simpson, 2016; Trias et al., 2016).

ELS is an end-to-end security model. Unlike many current architectures that use gateways, proxies, and application layer firewalls, ELS allows and encourages unbroken encrypted traffic between requester and service provider. The entity endpoints can directly authenticate each other and create an encrypted communication path from one authenticated entity to the other. This is in line with the idea of a zero trust architecture (ZTA). Security relies not on a secure network but on secure authentication and authorization by and at the endpoints.

ELS separates authentication from access and privilege. This separation runs counter to the common approach of using single sign-on (SSO) for both. The SSO approach simplifies the process, but it is a single point of failure. With ELS, a separate Security Assertion Markup Language (SAML) token is used for access and privilege, which is provided when requester attributes satisfy access rules.

ELS attempts to automate much of the access process. By defining access rules up front and maintaining attributes of all entities in the enterprise, access is quick to be granted and removed. As soon as the relevant rules or attributes change, access is dynamically recomputed to allow or deny access accordingly. This removes the current lag in gaining access and eliminates lingering access, which is the cause of many security problems.

ELS includes asynchronous messaging and content management solutions, as well as web-based requests. These solutions both rely on digital signatures by endpoints with PKI credentials. In these cases, the content is signed at the time of creation, but it might persist for long periods of time. Messages are typically consumed quickly, but their asynchronous nature means timeouts are much longer, if they exist, than with synchronous web traffic.

The main ELS issues for PKI are web-based authentication and digital signatures for content and messages.

3 NAMING ISSUES

PKI certificates are most meaningful if they use meaningful names. Naming of entities in an enterprise is therefore a critical first step in establishing a secure PKI. Typically, the certificate contains a name field and several attribute fields. The distinguished name (DN) encompasses both the name field and attributes, such as employee number or other unique identifiers, and satisfies the primary naming requirements of being unique and meaningful. First and last names are used as part of the DN to provide an immediate way for people to recognize each other. The additional information includes a unique identifier, which makes every DN unique within the enterprise.

If the enterprise has internal structure, this may be represented as additional fields in the DN. An employee's name and division may be included in the DN, along with the organization name and location. This helps to identify the individual by defining the scope of activities or where they work.

A decision needs to be made about which information is in the certificate and which information is retrieved from other sources based on the DN or other information in the certificate. For simplicity, the PKI certificate should contain only information relevant to establish the identity of an individual. Remaining attributes should be stored in authoritative content stores (ACSs) based on the DN as in the PKI certificate.

It is important that each ACS is the authoritative source of the attributes that it provides. It may be duplicated for performance, ease of use, or other reasons, but the duplicates must point back to the original source, and any updates to attributes are only authoritative when the single authoritative source is updated.

Text-based names require special consideration when they contain multiple parts. The X.509 standard (X.509, 1999-2012) addresses this issue through the format of the certificate, which includes separate identified fields with designated types and lengths. The DN itself has a type and length, and the components of it also have their own type and length. This prevents ambiguities that can arise through the use of simple text-based names.

However, the DN is often represented and used in text instead of its binary X.509 format. A one-to-one correspondence between X.509 format and other formats is necessary to preserve uniqueness. This often involves limiting the allowed content of the attribute fields to prevent ambiguities between delimiters and content.

4 NAME USE ISSUES

Establishing unique names and authoritative attribute stores is not enough for a viable PKI. The entities in the enterprise must actually use the certificates, data sources, and other PKI services effectively.

For a single enterprise with its own PKI, this is fairly simple. However, when interacting with the web or other partner enterprises with their own PKIs and trusted root CAs, name use can get more complicated.

For example, the U.S. Department of Defense (DoD) hosts its own set of root CAs and intermediate CAs. The root CAs sign the certificate of each intermediate CA, which may directly issue user certificates or continue the chain further to more intermediate CAs. The process of user authentication requires tracking this chain of CAs back to a trusted root CA, as shown for Alice and Bob on the left side of Figure 1. Chaining Alice's certificate to the trusted DoD root CA validates that the her identity can be trusted.

However, an interesting problem occurs when working with partner organizations. For example, the U.S. Department of Veterans Affairs (VA) has many current and former members of DoD, but it has its own root CA, which is issued by a non-DoD entity. If DoD and VA collaborate, VA users and servers are part of a separate PKI. The first step is to establish trust between these two PKIs. This allows the DoD to trust certificates issued by the VA CA.

A potential problem arises, however, if the VA creates a certificate with a DN that matches one within the DoD PKI, as shown by the duplicate certificate for Alice on the right of Figure 1. The problem is that the DoD trusts the VA CA and will trust the VA certificate, but the DN on the VA certificate matches a possibly different entity's account within the DoD. As a result, someone with such a VA DN would be able to access all the DoD resources that the corresponding DoD entity has access to.

An alternative situation is a rogue intermediate CA in either organization. A rogue intermediate CA could intentionally issue credentials with targeted DoD DNs in an attempt to access those entities' resources within DoD. For example, DoD CA #2 could issue Bob an additional credential with Alice's DN to provide Bob unauthorized access to Alice's resources.

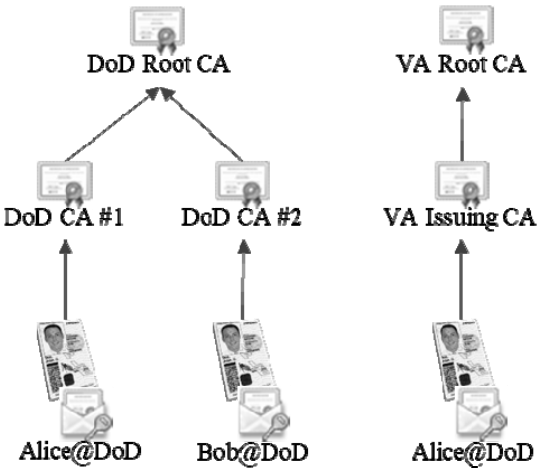

Figure 1: Name confusion across PKIs.

This highlights the importance of not only checking the full DN, but also the CA chain that the DN is part of. A DoD-style DN that is issued by the VA is highly suspect, and at the very least such a credential should be flagged and denied normal DoD access. Such measures could also be taken within DoD to protect identities from rogue internal intermediate CAs. More details of federation approaches based on trust levels is provided in (Foltz and Simpson, 2016c).

5 KEY GENERATION AND DISTRIBUTION ISSUES

There are two types of private keys in a typical PKI instantiation. The first is a signature or identification key, which is used for authentication or digital signatures. This is generated on secure hardware and has no copies or duplicates. The hardware protects the key from export so that there is only one copy in existence. This is useful for managing identities and non-repudiation. The entity associated with the private key is provided the only copy of the private key.

The second type of key is used for decryption, and this is typically archived so that encrypted data can be retrieved if the original decryption key is lost, destroyed, or expired. Generation and distribution for decryption keys is less secure than for identity or signature keys because the decryption keys have additional copies. Decryption keys can be generated in hardware and transmitted in encrypted form to other secure hardware, or they can be generated in or copied to software.

Identity and signature keys must have a single copy in order to preserve non-repudiation. This is generated in secure hardware and never leaves the hardware. The hardware provides an interface to use the key but does not provide a method for extraction. The Trusted Platform Module (TPM) is an approach that uses hardware-based key management (Trusted Computing Group, 2016), but current user key management solutions typically use portable external key stores, such as smart cards or USB/FireWire devices.

The challenge for modern systems is how to generate and use such keys on a mobile device. External hardware key stores, such as smart cards or USB/FireWire devices are convenient and secure for generation, but using them with mobile devices can be difficult. Integrated key storage and use within the device is more convenient.

The first thing needed for device-level key management is trusted hardware that is inseparable from the device itself. This hardware generates random key pairs and prevents the private key from being extracted. It makes the public key available. The private key can be used through a defined interface, such as PKCS11 for signature or authentication (Oasis, 2015). Such a hardware module is much like an attached smart card or USB stick with private key operations, except it is embedded inside the device. It is independent from the normal processor, and its internal structures are not made available except through defined interfaces.

As part of a PKI, such keys need certificates, which could be loaded into the hardware module or kept in normal storage. The challenge for the enterprise CA is to validate that keys are generated in such a secure hardware module. Smartcard or USB stick credentials are installed onto known custom hardware at manufacture time. Generating keys on a mobile device after a user has taken possession means that the state of the hardware is uncertain. For example, a malicious user may generate keys in software and claim that they were generated in hardware. This could be from a desktop, an application on the same mobile device, or a custom-built or compromised mobile device specifically designed to mimic the behavior of a normal phone while actually making private keys available. Alternatively, a hostile entity could manufacture devices designed to compromise the keys of users. If the device manufacturer is not trusted, it is not possible to have a trusted PKI with device-based keys.

With a trusted vendor, there are measures to stop fake devices. The vendor first registers each phone using an embedded public key that is installed at manufacturing time. This is a permanent feature of the trusted hardware and, hence, the device. The enterprise, when issuing mobile devices, validates that all such devices are listed in the vendor's registry and validates each device by performing a private key operation with the device.

The next step is to ensure that the public key presented for a certificate was generated on that device. This requires hardware support with a special operation. This is a signed assertion of key generation by the trusted hardware. It certifies that the secure hardware generated the provided public key and associated private key by providing a signature using the permanent embedded hardware key. The only way to generate such a signed statement is when the hardware generates and returns a new key.

This hardware operation ties the public key to a known device and allows hardware-based bootstrapping for further PKI development.

The public key associated with the device-based user credential is unique. As a result, the key generation process, which produced a device-permanent-key signature of the user credential public key, can be used with the registry of devices to confirm which device a user request is coming from. Combining this information with an attestation report from the device certifying that the device is in a valid state, the result is strong assurance that the user request is coming from the proper user on a device in a valid state.

An alternative approach would be to embed device information in the user credential. If the device public key is embedded in the user credential, a server can extract this and use it to verify a signed attestation report from the device.

The choice of whether to embed information in user credentials or provide a registry is up to the organization. Either solution provides a way to bind a user to a device for a given request by using the device-based user credential's unique public key. This process enables the enterprise to leverage the convenience of credentials embedded within mobile devices while preserving the hardware security properties associated with smartcards and other custom hardware key stores.

6 DIGITAL SIGNATURE ISSUES

When an entity signs digital content, it is intended to mean "I approved this content." However, digital

signatures in their raw form leave a lot of implicit real-world information out. For example:

- When was the content approved?
- For how long is the approval valid?
- For how long is the digital signature valid?
- What happens at credential expiration?
- What if the signing credential is revoked?
- What happens for role-based signatures when roles change?

These issues are not specific to digital signatures, but they are issues that can be more precisely addressed with a digital signature solution; thus agreement on which solutions to implement and how to implement them is important.

6.1 Time of Signature

The issue of when content is approved has a simple solution, which involves explicitly noting the date and time in the signed content. However, relying on the signing party to provide the date allows the possibility to pre-date or post-date documents.

To address this, a trusted third party validates the time of the signature. This can be implemented as an enterprise time service, and this service simply takes content, appends a time stamp, and signs the combined result. The original signer can then sign the combined (content + time stamp + time service signature) data to prove that the signature was done after the time in the combined data. The time service must be trusted to provide the proper date and time, and its issuing CA must be trusted by the receiver. To provide proof of signature before a certain time instead of after, the signed content can be presented to the time service for signature. For a time window, the time service can be called both before and after the original signer's signature is applied. This bounds the time of signature within the window between the two times.

For a multi-signature document, each signature can provide its own time window, a single window can be applied to the full set of signatures, or any combination of time stamps can be used throughout the signature process, depending on the desired level of granularity. Also, individual copies of the document can be signed by individual members with or without their own time stamps, and then the combination of document and all time stamped signatures can be time stamped upon aggregation. This might be desirable if the order of signatures is not important and parallelization is desired in the signature process due to time or communication constraints.

For signatures after a certain time, the time service need not sign the content of a document. Instead it could periodically provide a signed statement of the current time. Inclusion of this signed statement of time in the original signed content would be one way to show a signature occurred after a given time. However, this could not be used to show that a signature was performed before a given time.

6.2 Explicit Signature Validity

The validity time window of the signature is sometimes shorter than that of the credential. For example, a temporary authorization letter would likely expire while the signing credential is valid. In this case, the validity window can be included in the content to be signed. Because this is set by the original signer, no additional signatures are needed. This would generally apply to policy documents that are to be used to guide actions. Renewals are possible, but they would require a new signature.

Another issue is if the certificate associated with a document signature is revoked. The time of the signature may be available, but the time of revocation may not be provided in a validity check, and just the current status is reported. This raises the question of whether the document and its signature should be considered valid. Additional information may be of use in making this determination:

- When the revocation happened
- When the certificate first became invalid (looking retroactively)

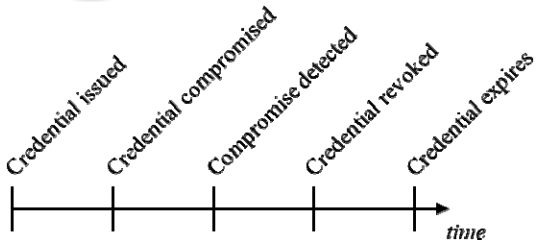

Figure 2: Credential compromise timeline.

The revocation time may be enough information to determine validity. This assumes that a valid certificate is actually valid and that the only factor in determining the validity of the signature is determining the validity of the certificate at the time the signature was performed. The issue with this approach is that revocations typically occur after

some sort of problem is discovered with the certificate, its associated key, or the entity it refers to. This provides a time period between the start of the problem and its discovery and eventual revocation action. An attacker could use such a time period to create unauthorized signatures that the enterprise would accept as valid.

As part of the revocation process, it may be possible to determine the point at which the credential was first considered invalid, which could be significantly before the revocation. For example, in the case of a missing or lost credential, the report might come in a week or so after the user last saw the credential. In this case, the validity window should end at the last time the user had possession of the credential, because any further use would be by someone else and not valid. In the case of a discovered insider attack, the certificate might be considered invalid at a much earlier time, depending on when the person was deemed to have turned against the organization. Figure 2 illustrates the different events on a timeline.

Implementing this requires the technical capability to record and make available the revocation time and validity time. It also requires the process of determining the validity time as a part of revocation. It requires the Online Certificate Status Protocol (OCSP) or a related protocol to allow queries and responses about the validity and revocation times. A simple modification would be to add a time to the OCSP request, which essentially asks, "Was certificate *C* valid at time *T*?" instead of just "Is certificate *C* valid?"

6.3 Signature Renewal

Signature renewal is generally used when a signing credential becomes invalid. This includes expiration or revocation. It may also involve changing roles and permissions for role-based signatures, where the position and authority, not the actual identity of the individual, is what matters. The signature by someone who changes roles may still be treated as valid, or it may be considered no longer valid. Also, the new person assuming the role may wish to explicitly invalidate previous signed documents even if they would otherwise be considered valid. Implementing such policies requires central services that maintain a list of signatures, their types, properties, and ties into a database of roles and their history. This builds on the basic PKI functions.

6.4 Signature Updates

Expiration is an issue when the signature outlives the signature certificate. Because cryptography can be broken, credentials are periodically replaced. At replacement key length may increase because existing techniques to break keys improve as computational capabilities increase. The cryptographic algorithms themselves may be changed as well.

Longer-term signatures can be broken if the underlying credentials are broken, so there is a natural limit on how long a signature is valid. Instead of simply indicating the validity period for the signature, more complicated measures must be taken so that old signatures are refreshed at a sufficient rate to prevent cryptographic or other attacks that are enabled by long periods of use.

Signatures provide a strong guarantee of identity at a particular time, but maintaining this over time is more difficult. It is generally not possible to construct a practical and scalable security measure that will survive for more than a few years, because technology advances and allows prior security measures to be broken more quickly each year. For this reason, central maintenance of official signatures may be required to maintain security over time. An automatic or manual re-signing may provide sufficient security, along with a history of all prior signatures.

The goal is to refresh old signatures using new keys or new technology before they become too old to be trusted as valid. Signed time stamps on the original and refreshed content are required to establish a proper chain of security from original signed content through to the current version. This would use the trusted time service mentioned above with a sequence such as the following:

Original signature:

1. Original content is time stamped (time service adds time and signature)
2. Time stamped content is signed by original signer
3. Signed content is time stamped a second time

Signature refresh:

1. Original triple-signed content (date, signer, date) is re-time stamped
2. Re-time stamped content is signed by same or new entity using updated keys/technology
3. Re-signed content is time stamped again

Further signature refreshing follows the same process as above. With this process, it is possible that invalid signatures exist for expired credentials, especially ones with broken cryptographic methods. It is important to monitor such aging signatures and generate new signatures while the current ones are still valid and secure.

The length of the signature chain grows over time, and it provides a verifiable history of the approval of the content. In addition, signed records of certificate revocations, role assignments, and other issues mentioned above must be maintained in order to tie the signatures to the real-world.

7 ELS SOLUTION FOR PKI

The ELS solution attempts to address the issues discussed. It starts with DNs for all users, established by a central authority. These include the name and unique identifying number in the CN and organizational information in other fields of the DN. The DNs are used in X.509 certificates that are issued after a full background check, proper authorization, and secure forms of identification are presented in-person at a credential issuing station. Certificates and associated keys are stored on smartcards for use with laptops or desktop machines.

For mobile devices, only approved vendor devices are supported, and they are enrolled in a device management system. The devices generate keys in a secure hardware enclave that is separate from the main processing unit and only provides defined interfaces for standard cryptographic operations involving the private key. A factory-installed and registered public/private key pair is used for attestation reports that certify the status of the device and its software. It also certifies the creation of a new public/private key pair in hardware.

Servers are provided a “security handler,” which is code to be executed upon receipt of an incoming request prior to application code. The security handler examines the user credential, and for mobile device credentials, it performs hardware validation of the associated device where the credential is stored, based on a registration of devices.

A time server provides two interfaces. One provides a signed statement of the current time. Another provides a signed statement that the given content provided by the user was received at the current time. This provides the ability to prove that a document was signed within a certain time window.

Validity windows for signed statements are provided by the signing entities. If the validity exceeds the credential validity, it must be renewed at expiration to remain valid. This may be by the original signer or by an authorized individual at the time of expiration. Formal signed documents are stored as official records and retain the full history of signatures.

One of the intentions for such a signature process is to allow automated enforcement of policy through PKI. If a valid signed statement (such as an XML structure) exists, then certain automated functions proceed; if not, they halt and produce alert messages.

In ELS, two-way, end-to-end PKI-based authentication is used for all web-based communication. The identity validation checks the full DN as well as the chain to a trusted root CA, and it validates that the DN values of each certificate in the chain are consistent with the issuing CA for that certificate. OCSP is used for certificate validation, and certificate revocation lists (CRLs) are periodically updated for backup to support operations when OCSP services fail or time out.

The web-based authentication portion of the ELS PKI solution has been subjected to two rounds of penetration testing. Other portions are currently in development, such as the mobile device credential management strategy and the higher granularity signature approaches. Current mobile device technology provides some native hardware capabilities for key management, but full integration into a working PKI on managed devices often requires work-arounds that use software instead of just hardware (Apple, 2019) (Samsung, 2019).

This work is part of a body of work for high-assurance enterprise computing using web services.

8 CONCLUSIONS AND FUTURE WORK

PKI is an integral part of the ELS solution for enterprise security. It forms the basis for many higher level security functions, such as authentication, authorization, and content integrity. This paper examines some of the challenges and shortcomings when using PKI and provides methods to use and extend the core PKI functionality to provide desired enterprise functionality.

Future work includes the extension of signature revocation and validity times to normal business use. For example, if signatures by certain individuals are

only to be considered valid if executed during their working hours or their shift hours, then a service can be invoked to determine for a given signature time whether the signer is authorized to sign. This is important when signature credentials are associated with the person's role or location. Another variant is to disable credentials when the owner is known to be on vacation, traveling, or otherwise in a position without the credentials. Such irregular intervals could be handled in a way similar to normal business hours using an external server to manage schedules.

The signature refreshing process, where the number of signatures grows over time and each signature encapsulates the prior ones, bears some resemblance to a blockchain approach, such as that used by Bitcoin (Bitcoin, 2019). The Bitcoin blockchain has a similar problem that current hashing and public key algorithms used for transactions may be compromised later, and algorithms and key sizes may change in the future. This suggests that a blockchain may provide a natural solution to the central signature repository. However, the Bitcoin blockchain is decentralized and requires vast computing resources to maintain, and such a solution is not desirable for an enterprise. Various private blockchain technologies attempt to resolve this issue, but by making the blockchain private, they give control to a central authority and negate many of the features of the public blockchain. Also, although many signatures are intended for broad audiences, some are not, and a blockchain approach that does not encrypt the content appropriately is not an appropriate solution for general digital signatures. Hence, current blockchain technologies offer an approach that parallels some of the concepts of digital signatures, and further work in this area might provide a viable approach.

REFERENCES

Apple, "iOS Security, iOS 12.1", November 2018, https://www.apple.com/business/site/docs/iOS_Security_Guide.pdf

Bitcoin, "Blockchain," Bitcoin Developer Documentation, available at https://bitcoin.org/en/blockchain-guide, accessed December 9, 2019.

Cooper, D., et al., "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", May 2018. Available at https://tools.ietf.org/html/rfc5280, accessed December 9, 2019.

Entrust Datacard, "What is PKI?" available at https://www.entrustdatacard.com/pages/what-is-pki, accessed November 27, 2019.

Foltz, K. and Simpson, W., 2017. Enterprise Level Security with Homomorphic Encryption. In *Proceedings of 19th International Conference on Enterprise Information Systems (ICEIS 2017), Porto, Portugal, April 26–29, 2017*.

Foltz, K. and Simpson, W. R. 2016. "The Virtual Application Data Center." In: *Proceedings of Information Security Solutions Europe (ISSE) 2016. Paris, France*.

Foltz, K. and Simpson, W. R. 2016. "Enterprise Level Security – Basic Security Model." In: *Proceedings of the 20th World Multi-Conference on Systemics, Cybernetics and Informatics: WMSCI, Volume I, WMSCI 2016. Orlando, FL*.

Foltz, K. and Simpson, W. R. 2016. "Federation for a Secure Enterprise." In: *Proceedings of The Twenty-first International Command and Control Research and Technology Symposium (ICCRTS 2016). London, UK*.

Oasis, "PKCS #11 Cryptographic Token Interface Base Specification Version 2.40," Oasis Standard, April 14, 2015. Available at http://docs.oasis-open.org/pkcs11/pkcs11-base/v2.40/os/pkcs11-base-v2.40-os.doc, accessed December 9, 2019.

Samsung, "KNOX Platform Security", Samsung Developers website, Available at https://developer.samsung.com/tech-insights/knox/platform-security

Simpson, W. R. 2016. *Enterprise Level Security – Securing Information Systems in an Uncertain World.* Boca Raton, FL: CRC Press, p. 397.

Trias, Eric D., et al. 2016. "Enterprise Level Security", *Proceedings of the 35th MILCOM conference*, DOI: 10.1109/MILCOM.2016.7795297 pp. 31-36, http://ieeexplore.ieee.org/document/7795297/.

Trusted Computing Group, "TPM 2.0 Library Specification", September 29, 2016. https://trustedcomputinggroup.org/resource/tpm-library-specification/

X.509 Standards

a) DoDI 8520.2, Public Key Infrastructure (PKI) and Public Key (PK) Enabling, 24 May 2011
b) JTF-GNO CTO 06-02, Tasks for Phase I of PKI Implementation, 17 January 2006
c) X.509 Certificate Policy for the United States Department of Defense, Version 9.0, 9 February 2005
d) FPKI-Prof Federal PKI X.509 Certificate and CRL Extensions Profile, Version 6, 12 October 2005
e) RFC Internet X.509 Public Key Infrastructure: Certification Path Building, 2005
f) Public Key Cryptography Standard, PKCS #1 v2.2: RSA Cryptography Standard, RSA Laboratories, Oct 27, 2012
g) PKCS#12 format PKCS #12 v1.0: Personal Information Exchange Syntax Standard, RSA Laboratories, June 1999.