TRUST-BASED SECURE POSITIVE TRAIN CONTROL (PTC) INTEROPERATION

Mark Hartong[1]
Department of Information and Software Engineering, George Mason University, Fairfax, VA 22030
Office of Safety, US Federal Railroad Administration, Washington, DC 20590.
mhartong@gmu.edu
mark.hartong@fra.dot.gov

Rajni Goel
Department of Information Systems and Decision Sciences, Howard University, Washington, DC 20059.
rgoel@howard.edu

Duminda Wijesekera
Department of Information and Software Engineering, George Mason University, Fairfax, VA 22030.
dwijesek@gmu.edu

ABSTRACT

Positive Train Control (PTC) is a wireless control system ensuring railroad safety by enforcing train separation, speed enforcement, roadway worker protection and other safety functions. Due to shared track rights over each-other's tracks in North America, company A's trains must be safely operated by company B's crew on company C's tracks, requiring different PTC systems to securely interoperate with each other. For a security framework to ensure that, we propose using a trust management system with certificates and over the air re-keying (OTAR). Back of the envelope calculations show that our solution meets timing needs of PTC.

Index Terms: Security, Rail Transportation Control, SCADA Systems, Cryptography

[1] The views and opinions expressed herein are those of the authors and do not necessarily state or reflect those of the United States Government, the Department of Transportation, or the Federal Railroad Administration, and shall not be used for advertising or product endorsement purposes

1. Introduction

Positive Train Control (PTC) Systems [1, 2] (Figure 1) provide inter-train separation, speed enforcement, and roadway worker protection utilizing wireless communications to exchange control information. In order to do so, on board systems and locomotive crews communicates with office dispatchers and wayside devices.

The business model that has evolved in North America, supported by legislation and commercial agreements, allows railroad companies to cross-travel on each other's tracks by exchanging power, locomotives and crew. This gives rise to the central operational use case for American railroads: namely, that an authenticated crew from railroad A should be able to safely operate an authenticated locomotive from railroad B and can safely travel over an authorized shared track segment belonging to railroad C. As an interoperable solution that enables the necessary security with the requisite safety, we propose using a distributed trust management system coupled with a PTC-system. The paper consists of a technical analysis of operational requirements, possible misuses of enforced requirements and operational infrastructure, and a distributed trust management system that implements the central use case while preventing selected misuse cases.

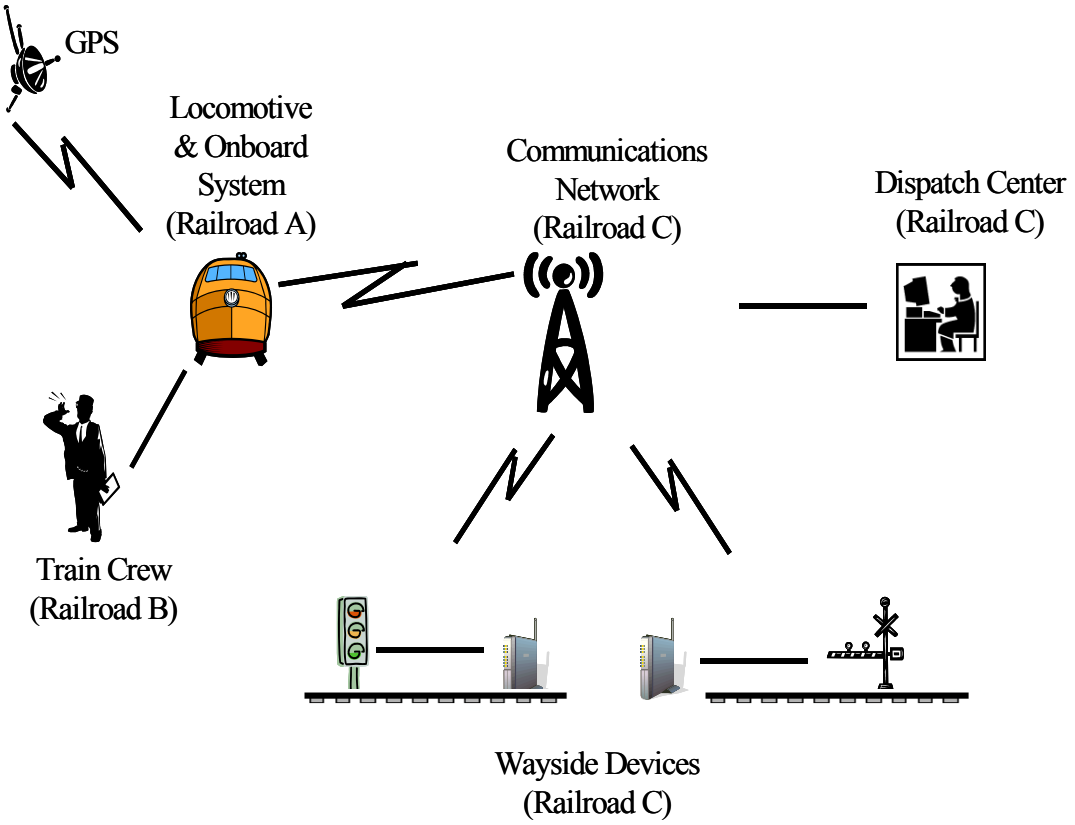

Figure 1: An Interoperable PTC System

The main misuse cases we consider in PTC inter-operation are impersonation and claiming invalid track rights. For example, a crew or a locomotive, claiming to be from railroad A or railroad B respectively, may not be authentic, and the combination may not be authorized to enter a segment of company C's railroad. This situation constitutes a platform independent misuse case. A second misuse case arises from exploiting the vulnerabilities of the underlying (wireless and wired) communication infrastructure and protocols, by dropping, delaying, altering or introducing packets. Compromising safe PTC inter-operations using network level mal-activity constitutes a platform dependent misuse case. As will be shown shortly, our proposal prevents the platform independent misuse case and detects the occurrence of the platform dependent misuse case while enabling the central use case.

The rest of the paper is organized as follows. Section 2 summarizes the major components of a distributed trust management system. Section 3 describes its application to PTC interoperation and OTAR for key distribution. Section 4 describes performance characteristics of proposed solution, and Section 5 concludes the paper.

2. Trust Management for PTC Systems

The main utility of a trust management system (TMS) is distributed authentication and authorization without a central authority. It does so by using three primary components: *certificates* carried over *distribution protocols* that enforce *policy* decisions. In a typical use case, each service requestor, being issued a certificate (an un-forgeable statement issued by a, Certificate Authority (CA)), stating that selected attributes have specific values) presents the certificate to obtain service and furnish proof of authenticity. The service provider verifies its authenticity and the validity with the issuer prior to granting the service. That requires every user to belong to a trust domain represented by a certificate authority. All service requesters are authenticated and service level agreements (SLAs) are cross checked at service time.

Applied to the Use Case described earlier, when, a crew (claiming to be from company A) arrives on a locomotive (claiming to be from company B) wishes to enter a segment of company C's tracks, the entry checkpoint to C's tracks is presented with certificates of service entitlements and authenticity for A's crew issued by A's CA and B's locomotive issued by B's CA. The checkpoint presents these certificates to its own CA who cross checks them with A's and B's CAs respectively. If entities are authenticated and certificates are verified to be valid, and upon having a valid track rights, B's locomotive operated by A' crew are granted entry onto C's tracks. Thus, this solution, if enforced as claimed, would evade the first misuse case.

Digital signatures are byte strings with keys - that are used to (en/de)-crypt information. The key management certificates specify the details about the keys, their validity period and how they can be

updated etc. Certificates, consisting of values of chosen attributes are digitally signed by CAs to validate integrity. Similarly, all requests and responses are signed to evade alteration and injection of spurious messages. This detects the secondary misuse case.

2.1. Certificates for PTC Interoperation

The certificates issued to authenticate and authorize crews, on board systems, wayside devices, and dispatchers follow the standard X.509 public key data structure [3] that binds the subject's public key and privileges to their identity via a X.509 signature (DSS) certificates and a X.509 key management (KEA) certificates. The X.509 attributes pertinent to enforce secure PTC interoperation are as follows:

1. **Version (Type: integer)** Purpose: enable parsing
2. **Issuer (Type: integer)** Purpose: issuer ID
3. **Name (Type: integer)** Purpose: user ID
4. **Seril_number (Type: integer)** Purpose: cert. ID
5. **Signature (Type: integer,algorithm name)**

   Purpose: Algorithms used for signing and hash
6. **Validity Period (Type: time interval)**

   Purpose: (begin, ending) time of certificate validity

While other fields are self explanatory, the (*issuers ID, serial_number*) pair provides a system-wide unique ID for a certificate. In order to manage certificates, the PTC system has to create a key management infrastructure and policies

```
Certificate:
  Data:
    Version:  3
    Serial Number: 1
    Signature Algorithm: md5withRSAEncryption
    Issuer: C=US, ST=KS, O=BNSF OU=Signal, CN=CA
    Validity: Not Before 1 July 2006 0001Z
              Not After   1 July 2007 0000Z
    Subject Name; C=US, ST=KS, O=CSX, OU=OPS, CN=Casey Jones
    Public Key Info
      Algorithm:  RSA Encryption
      RSA Public Key: <Public Key>
    X.509 v3 Extensions
      Basic Constraints: Critical
      Basic Constraints Shared Authorization
        Train Owner: C= US ST= KS O= BNSF; C= US, ST=NE, O=UP
        Train ID: OU=OPS, CN=1234; CN=W3F4; TY65
        Track Block: Anna, Bess   Subdivision: Beardstown  RR: BNSF
        Time: Not Before 1200Z 24 Aug 2006
              Not After   1200Z15 Oct 2006
  Signature Algorithm:
```

Figure 2: X.509 v3 Certificate for SLA

Figure 3 shows a sample certificate identifying its holder as engineer Casey Jones of the CSX Railroad, and stating BNSF CA will verify his identity for the period 1 July 2006 to 1 July 2007. This certificate authorizes engineer Jones to operate trains owned by either UP or BNSF railroad, with Train Symbols 1234, W3F4, TY65 on the Anna and Bess blocks of the Beardstown Subdivision of the BNSF during the period 24 August to 15 October 2006, stated in the *part X.509 v3 Extensions*. This certificate is constructed to be furnished by the locomotive to the office dispatcher of a company. The certificate recipient must verify its validity with BNSF's CA, and is at liberty to demand authenticity proof from Casey Jones and verify that with CXS's CA, or accept BNSF's verification.

2.2. Example Policy Attributes

The following represent example attributes chosen for a trust management system.

1. Admission to a trust domain:

- Procedures for physical identity validation
- Physical identity-to-key binding
- Registration attributes

2. Initializing cryptographic material:

- Key generator and-cryptographic algorithms
- Configuring algorithmic preferences
- Identification of trusted parties
- Definition of domain and-trusted parameters.

3. Key management:

- Installation, establishment, key generation and distribution of active keying material.
- Key update process, transporting initial keys and crypto periods.
- Archive & Recovery: storage and protection.
- Accountability*:* Access requirements to keying material generating-distributing-destroying-and conducting audits.

4. Survivability of Critical Infrastructure:

- Continuity of operations plans
- Backup and recovery mechanisms for trust breaches and system-wide failures.

The following illustrates a policy for our PTC truest management infrastructure.

1. Admission: An example PTC admission control policy is role-based access control (RBAC) where every user is granted minimal access to play its role.

2. Authentication: Physical subjects (Onboard, Wayside, Office/Dispatch) are bound to their public key with a certificate and a hardware level address (such as the MAC address). Humans are required to use a combination of Biometric information, passwords and PINs.

2. Initializing cryptographic material: These include the key generation algorithms for private/public key pairs such as RSA, seeds and algorithmic preferences such as AES for encryption and, MD5 for hashing.

3. Key Management: Uses a PKI infrastructure and communicates original keys through an out of band secure channel. Archive and Recovery can require a user to replace elapsed certificates with current ones. Accountability is enforced through certificate revocation lists.

4. Survivability: In the event of a disaster, CA operations are reestablished first. If the CA is physically damaged with copies of signatures, keys and certificates are replaced.

2.3. Service Level Agreements

Service level agreements containing track right using X.509 certificates are enforced. The Interstate Commerce Act authorizes the Surface Transportation Board of the US Department of Transportation to impose conditions (between railroad companies) on the granting of reciprocal switching, track access provisions, and reciprocal access to facilities. The acquisition of tracking rights by a rail carrier over a railroad line operated by another rail carrier requires prior Board approval under 49 U.S.C. 11323. Once the SLA has been negotiated and approved it can be enforced using certificates an authentication as discussed earlier.

2.4. OTAR

We propose using Over-the Air-Rekeying (OTAR) [4, 5, 6, 7] for keying. OTAR is a suit of protocols in use with the US Department of Defense and specified for use in TIA/EIA Project 25; it protects inadvertent key disclosure and limited unauthorized modification.. OTAR has valid crypto periods, and operational procedures including *key management changeover message* sent from the KMF/KDC to end units. OTAR uses Traffic Encryption Keys (TEKs) and Key Encryption Keys (KEKs). TEKs are used to hash data traffic and KEKs are used to encrypt and exchange other KEKs or TEKs.

3. Secure PTC Interoperation

Security of PTC interoperation is ensured by enforcing authentication, authorization and SLAs.

3.1. Authentication

Railway-engineers and physical equipment are required to be authenticated in order to request or provide service. Static equipment such as wayside devices and those at dispatcher offices always authenticate themselves to the domain controller in their own domain. When an engineer or a locomotive starts from a domain outside of their own, the local domain controller can verify the authenticity by contacting the appropriate CA for assistance.

The authentication process is known as the Modified Needham-Schroeder protocol and is illustrated in Figure 3

$A \rightarrow B : P_B(k_1 : A)$
$A \leftarrow B : P_A(k_1, k_2 : B)$
$A \rightarrow B: P(k_2)$

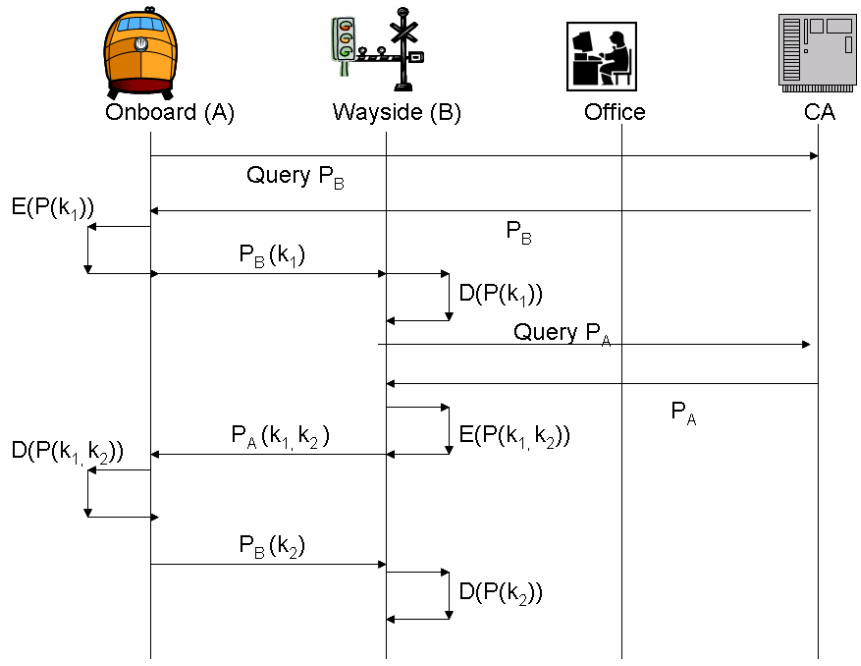

Figure 3: Needham-Schroeder Authentication

The service requester A encrypts a session key k_1 using the public key of B and submits it to the service provider B. By first checking a certificate revocation list (CRL) to ensure that B's public key is still valid, using the service provider B's public key, A is assured that by using B's public key only an entity holding B's private key will be able to determine the value of the k_1. Upon receipt of the message from A, B decrypts the message using its private key. Similarly, after checking the CRL to ensure the validity of A's public key, B encrypts the received k_1 and an additional key k_2. Upon receipt of the message from B, A decrypts the message and validates that the received k_1 matches the transmitted k_1. A then encrypts and transmits k_2 back to B, where B validates it matches the nonce k_2 it transmitted to A. At this point the service provider and the service requester have mutually authenticated and may proceed to authorization.

3.2. Local Authorization

At the time of security service design, based on the roles played by various subjects, they are issued keys by the KDC and certificates by the CA specifying attributes necessary to obtain services, such as the authorization certificates issued to engineer Casey Jones shown in Figure 3. In order to obtain access to a resource (or require service) the requester presents the service request to the service provider. For a local request, after establishing the requester's authenticity, the service requester presents the certificate to the local CA and verifies its validity. If valid, grants the service.

R, certificate $_A(a_1, a_2, \ldots a_N)$: (hash (k_1 k_2): A)

A → B : (R, certificate $_A$: (hash (k_1 k_2): A)

IF (R, certificate $_A$: (hash (k_1 k_2): B) = (R, certificate $_A$: (hash (k_1 k_2): A)

THEN WITH ACDF

FOR R AND certificate $_A$ (a_I): A) *AND* (A! permitted_service$_B$ (b_J): B)

IF (R) ε { certificate $_A$ (a_I): A} *AND*

(certificate$_A$ (a_I):A) ε {A! permitted_service$_B$ (b_J): B}

THEN A←B: (b_N = R)

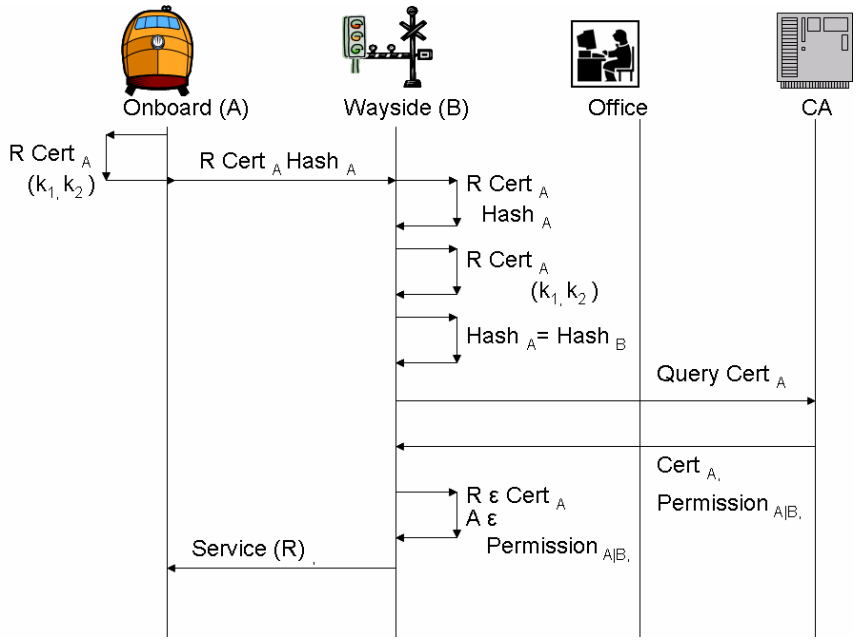

Figure 4: Authorization with Local ACDF

Figure 4 illustrates the authorization process with a local access control decision function (ACDF). The service requester first hashes the message containing the requested service(s) and their certificate containing their authorizations using a hash function with the combined cryptographic session keys $k_1k_{2.}$, appending the resulting hash value to their certificate and service request. The service request, certificate and hash are submitted to the service provider B. Upon receipt, the service provider splits the received service request and certificate from the appended hash. Using their copies of the combined cryptographic keys $k_1k_{2.}$ and the received service request R and certificate, the service provider computes a received hash value. If the received hash value matches the computed hash value, B knows that the service request and certificate was received unaltered.

The ACDF of B then examines the service request R and each authorization in the X.509 v3 certificate extension field received from A. If the requested service R is one of the set a_I authorized A which is also an element of the set of permitted services that B may provide to A then B provides the service to A. The set of permitted services that B may provide to a requester is obtained from B's CA, and depends upon the security policy of the system and the individual capabilities of the particular B.

3.3. Distributed Authorization using Service Level Agreements

If a service requester pair such as an engineer from railroad A on board railroad B's locomotive requesting entry to railroad C's tracks present a SLA encoded as a certificate requesting permission the distributed authorization process is similar to the local authorization process. Unlike before, A, B, and C are in different domains. Each entity, A and B, submits their individual service requests R_A and R_B to C along with their individual certificates. Each service request and certificate is hashed with their respective cryptographic session key pair k_Ak_C or $k_Bk_{C.,}$ Upon receipt, and after verification of the hash, the CA on domain C verifies the authorizations of A and B by contacting the appropriate CAs as it checks for the validity of the presented certificates from A and B. Once the authorizations are received from the appropriate CA, the ACDF evaluates if the service request R_A and R_B are elements of the respective authorizations of A and B (indicating that A and B have authorization for the requested function) and if the requested service pair R_A R_B is both authorized to be provided by C and can be provided by C. Multi-domain authorization is illustrated in Figure 5.

3.4 Satisfying Use Cases and Preventing Misuse Cases

The central use case that A's crew on B's train must be allowed on a section of C's track is satisfied while avoiding all misuse cases using the steps of authentication, authorization exchange, and validating the SLA.

R_A, certificate $_A(a_{1A}, a_{2A}, \ldots a_{NA})$: (hash ($k_A$ k_C): A)

R_B, certificate $_B(a_{1B}, a_{2B}, \ldots a_{NB})$: (hash ($k_B$ k_C): B)

A → C : (R_A, certificate $_A$: (hash (k_A k_C): A)

B → C: (R_B, certificate $_B$: (hash (k_B k_C): B)

IF (R_A, certificate $_A$: (hash (k_A k_C): C) = (R_A, certificate $_A$: (hash (k_A k_C): A)

AND IF

(R_B, certificate $_B$: (hash (k_B k_C): C) = (R_B, certificate $_B$: (hash (k_B k_C): B)

THEN WITH ACDF

FOR R_A AND certificate $_A$ (a_{IA}): A) *AND* (A! permitted_service$_C$ (c_J): C)

IF (R) ε { certificate $_A$ (a_I): A} *AND*

(certificate $_A$ (a_I):A) ε {A! permitted_service$_C$(c_J): C} *AND*

FOR R_B AND certificate $_B$ (a_{IB}): B) *AND* (A! permitted_service$_C$ (c_J): C)

IF (R) ε { certificate $_B$ (a_{IB}): A} *AND*

(certificate $_B$ (a_{IB}):A) ε {A! permitted_service$_C$(c_J): C} *AND*

IF R_A•R_B ε (A X B! permitted_service$_C$ (c_J): C)

THEN A←C: (b_N = R_A)

THEN B←C: (b_N = R_B)

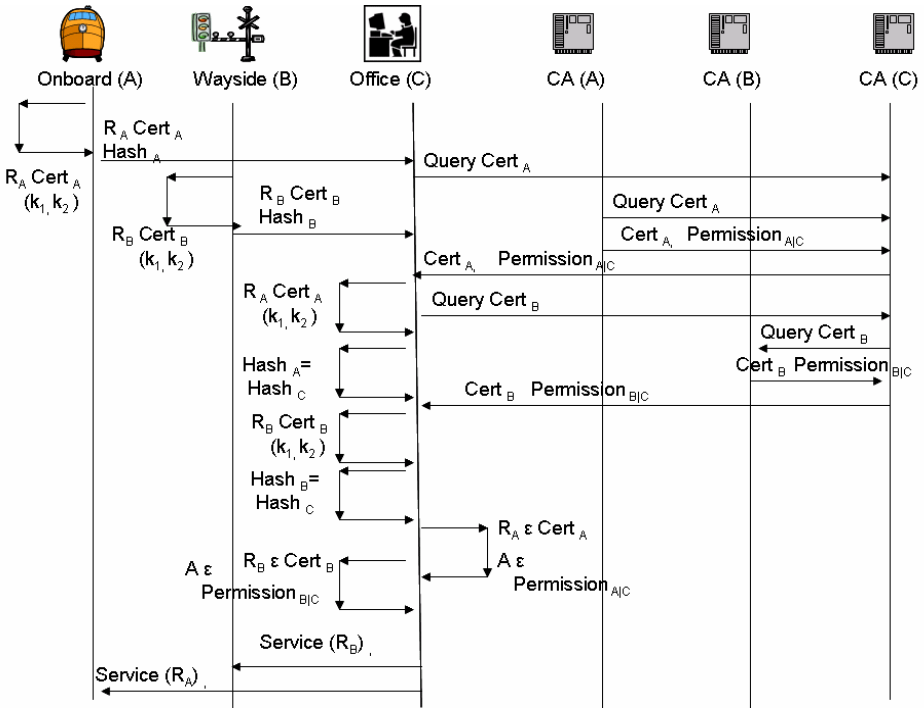

Figure 5: Multi-Domain Authorization

The steps of authentication, authorization, and validation, while imposing an overhead on the central use case, do not prevent it. These same steps, however, prevent the primary and detect the secondary misuse case.

As shown in the previous section, authentication and distributed authorization ensures that the engineer and the locomotive are properly authenticated and they carry a valid SLA. Thus, the primary Misuse Case of identity fraud or the SLA misuse is prevented, while enabling the central Use Case.

While it is the act of signing and/or hashing exchanged messages that actually prevents the successful execution of both the primary and secondary misuses, OTAR provides for the necessary certificates and keys exchange to sign and hash, therefore preventing the secondary misuses. Message alteration is detectable (but not preventable) by receipt of an invalid hash, while spurious messages can be detected by receipt of an invalid signature. Message drops are detected by having a message number hashed. Message delays are detectable by comparing the sent times and transmission delays, but not preventable by proposed cryptographic techniques.

4. Performance

Using a distributed trust management system to prevent or enable detecting misuse cases results in the following kinds of overhead:

1. Protocol overhead due to additional challenge-response steps.
2. Packet-size increment due to encryption headers
3. Packet size increment due to padding
4. Processing delays of cryptographic algorithm.

We now describe their affect.

4.1. Protocol Overhead

The protocol overhead imposed by a strong mutually authenticating challenge response protocol can be as low as three steps per challenge-response. Conceptually this appears to have little impact on the overall performance of the system. However, even though the number of conceptual steps may be small, the

actual time to execute the challenge response may be significant. In a worst case scenario, a public-private key encryption based schema is used with the participating entities not in possession of the public key of the other party, the public keys are issued by different CA's in different domains, and the keys are stored in chained hierarchal distributed directory information trees. The access time is equal to the $\sum T_I$, where T_I is the individual response times to a directory request for information and the transmission times between directories.

Potentially a directory query could require chaining up through multiple distributed directories to the root directory of the domain, across several domains, then down through multiple distributed directories to the appropriate node directory before the proper key is found. This process would be required twice if neither entity held the public key of the other. The transmission time of the certificate between directories is a function of the network bandwidth available between the directories, and the number of bits of data that must be exchanged. The depth of chaining required is a function of the directory structure, and the response time to a directory query a function of the processing capability of the individual directory components. Because of the potential adverse impact that the physical implementation may have, careful consideration must be given to the design of the system to minimize the impact on system performance.

The other impact that the challenge–response protocol has is based on the number of times that the challenge response is executed. Again, assuming a worse case scenario where the challenge-response were done for every block of data exchanged between two communicating entities, and assuming that each step of the challenge and response required 1 block, and assuming that a steady state has been reached (i.e. the appropriate public keys have been located and provided to the correct communicating parties), an efficiency no greater than 25% could be expected (1 block of data for 3 blocks of overhead). In practice the actual efficiency would be less.

4.2. Size Increment of Encryption Headers

For example, the IPsec [8] uses its own headers in addition to IP headers. IPSec - tunnel mode adds a new 20-byte IP header in front of the transported IP packet, and the IPSec- ESP mode adds an additional 8-byte ESP header, a 0 to 16-byte Initialization Vector (IV), and a 16-byte ESP Trailer. IPSec-AH adds a 24-byte AH header. The result is significant overhead related to the various modes as shown in Table 1.

Packet Size	Transport Mode ESpP3DES/SHA-1	Transport Mode ESP AES/SHA-1	Transport Mode AH SHA-1	Tunnel Mode ESP 3DES/SHA-1	Tunnel Mode ESP AES/SHA1	Tunnel Mode AH SHA-1
46	61%	70%	43%	104%	113%	87%
512	5%	6%	4%	9%	10%	8%
1500	2%	2%	3%	3%	3%	3%

Table 1: Worst Case Network Overhead

4.3. Packet Size Increment due to Padding

Common symmetrical encryption/decryption (DES, 3DES, AES) [9, 10] and hashing (SHA, MD5) [11, 12] algorithms uses data block, and consequently, pads packets before encrypting them. For example, SHA-1 and MD5 require 512-bit blocks (Block Size (Bits) 64 bytes). For randomly sized packets, padding as a percent of throughput increases as packet size decreases. Table 2 show the overhead of known algorithms, where the percentages are calculated based on smallest packet size that is greater than size in column header and divisible by required block size.

Algorithm	Packet Size	Small Packets	Average Packets	Large Packets	Packets Divisible
DES/ 3DES	40	15 %	2 %	0.5 %	0 %
AES	128	23 %	4 %	1 %	0 %
SHA-1/ MD-5	512	49 %	14 %	4 %	0 %

Table 2: Impact of Worst Case Padding

4.5. Processing delays in Symmetric vs. Asymmetric Algorithms

The overhead of asymmetrical cryptographic scheme depends upon the size of the asymmetrical KEK and the duration of the symmetrical TEK. The scenario given in Figure 6is used to compute the overhead delays in using asymmetrical cryptography [13], and the advantage of combining symmetrical and asymmetrical cryptography.

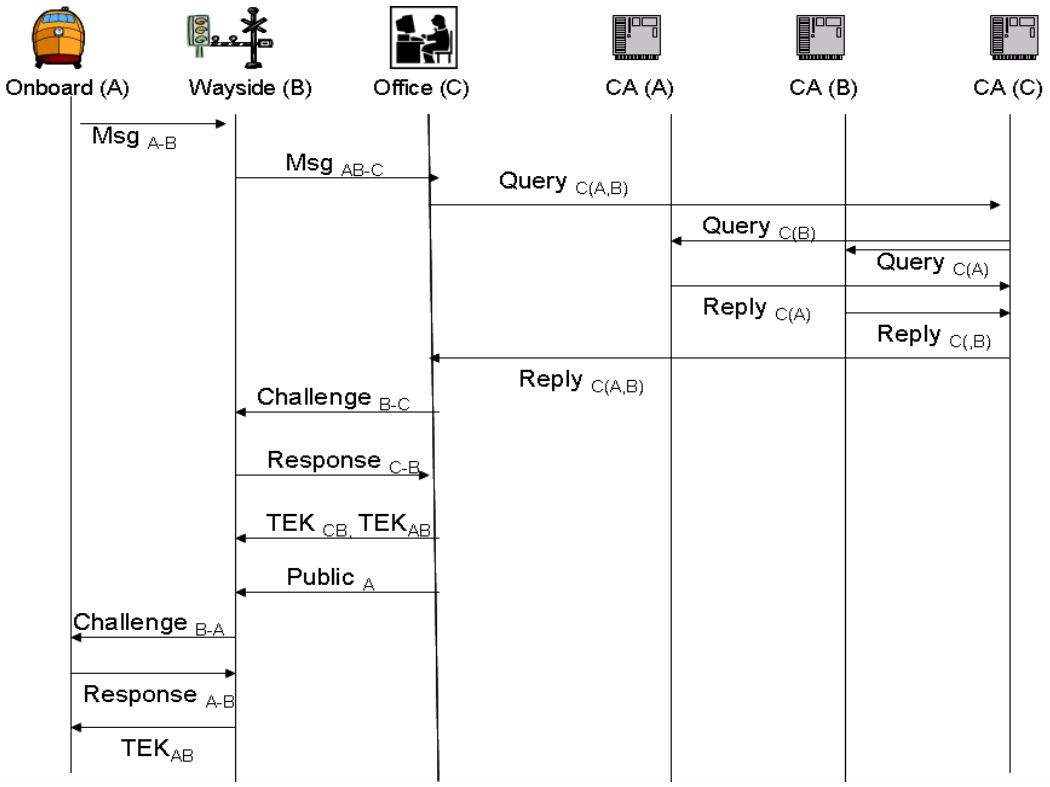

Figure 6: PTC Interoperation Example

To provide the same level of security using the same message size, asymmetrical cryptography is ~1000 times slower than symmetrical cryptography, although the exact number depends on specific algorithms and devices. Table 3 shows an *order of magnitude* estimate of relative cost to authenticate A using symmetrical and asymmetrical cryptography, where t and q are respectively the time for local processing and distributed processing with remote CA's. .

Activity	**Asymmetrical Encryption**	**Symmetrical Encryption**
A message to B	1000t	t
B message to	1000t	t

Activity	Asymmetrical Encryption	Symmetrical Encryption
C C queries CA	1000q	q
C CA C queries	1000q	q
CA A &B CA B CA A & CA B respond to CA C	1000q	q
CA C respond to C	1000q	q
C Challenge o B	1000t	t
B Respond C	1000t	t
TEKs C to B	1000t	t
Pub Key A from C to B	t	t
B challenge A	1000t	t
A respond B	1000t	t
TEK B to A	1000t	t
	4000q+8001t	4q+9t

Activity	Asymmetrical Encryption	Symmetrical Encryption
		Δ= 3996q+7992tt

Table 3: Comparative Time Costs

Table 4 substitute values obtained by (en/de) crypting random 65536 byte blocks using the Gnu C functions running under Fedora Core 5 on a 1.7 GhZ Celeron Processor. Used sizes are comparable to those by Yu [14].

Algorithm	Encrypt Speed	Decrypt Speed
DES	33.7 Mb/s	32.5 Mb/s
3 DES	20.6 Mb/s	18.6 Mb/s
AES (128)	76.7 Mb/s	74.4 Mb/s
SHA-1	32.3 Mb/s	-
MD-5	123 Mb/s	-
RSA	0.45 Mb/s	0.041 Mb/s

Table 4: Comparative Performance

5. Summary

Positive Train Control systems are been designed to decrease railway accidents by electronically enforcing speed restrictions, inter-train separation and a host of other requirements. Nevertheless, due to potential misuses of such a system, some security mechanisms need to be enforced upon their design.

We have proposed a rudimentary system that uses distributed trust management to ensure distributed authentication and authorization and OTAR for *online* key exchanges. They result in timing and processing overheads that need to be considered during the design stage of such a system.

Designing an effective security solution to PTC requires analyzing its strength, performance and cost against potential risk. If appropriately chosen, and when considered in light of organizational and environmental factors, a combination of managerial, operational, and technical controls can synergistically work together to ensure safe and secure interoperable PTC systems.

6. References

1. Federal Railroad Administration, ”Report of the Railroad Safety Advisory Committee to the Federal Railroad Administrator, Implementation of Positive Train Control Systems” Washington DC, August 1999

2. Federal Railroad Administration, ”Railroad Communications and Train Control”, Report to Congress, Washington DC, ,July 1994

3 ITU-T X.509/ ISO/IEC9594-8: 2001 “Information Technology-Open Systems Interconnection—The Directory: Public Key and Attribute Framework”, International Organization for Standards

4. TIA/EIA/TSB-102.AACA ”Over the Air Rekeying Protocol”, Telecommunications Industry Association

5. TIA/EIA/TSB-102.AACA-1“OTAProtocol”, Telecommunications Industry Association

6 .TIA/EIA-TSB-102.AACB “OTAR Operational Description”, Telecommunications Industry Association

7. TIA/EIA-TSB.AACC-2 “Conformance Test for the P-25 OTAR Protocol”, Telecommunications Industry Association9.

8. RFC 2401 Security Architecture for the Internet Protocol (IPSEC), Nov 1998

9. NIST Pub 46-3, DES, Oct 1999

10 NIST Pub 197, Advanced Encryption Standard, Nov 2001

11. NIST Pub 182-2 Digital Signature Standard, National Institute of Standards, 1 Aug 2002

12. RFC1321, “The MD5 Message-Digest Algorithm”, April 1992

13. RFC2313 PCKS #1 RSA Encryption Version 1.5

14. Yu, “Timing Test”, htttp://www.winlab.rutgers.edu/~yu/sub_pages/timingtest.htm