National Institute of
Standards and Technology
U.S. Department of Commerce

NISTIR 7539

Symmetric Key Injection onto Smart Cards

David A. Cooper
William I. MacGregor

NISTIR 7539

Symmetric Key Injection onto Smart Cards

David A. Cooper
William I. MacGregor

COMPUTER SECURITY

Computer Security Division
Information Technology Laboratory
National Institute of Standards and Technology
Gaithersburg, MD 20899-8930

December 2008

U.S. Department of Commerce

Carlos M. Gutierrez, Secretary

National Institute of Standards and Technology

Patrick D. Gallagher, Deputy Director

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analysis to advance the development and productive use of information technology. ITL's responsibilities include the development of technical, physical, administrative, and management standards and guidelines for the cost-effective security and privacy of sensitive unclassified information in Federal computer systems. This Interagency Report discusses ITL's research, guidance, and outreach efforts in computer security, and its collaborative activities with industry, government, and academic organizations.

National Institute of Standards and Technology Interagency Report
20 pages (2008)

Certain commercial entities, equipment, or materials may be identified in this document in order to describe an experimental procedure or concept adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the entities, materials, or equipment are necessarily the best available for the purpose.

Abstract

This paper describes architectures for securely injecting secret keys onto smart cards. Specifically, this paper details key injection architectures that could be implemented on Personal Identity Verification (PIV) Card in future. The primary goal is to create additional opportunities for the use of the PIV Card in Physical Access Control Systems (PACS). There is significant interest in conducting a fast, accurate, and highly secured authentication transaction using symmetric keys in PACS environments. This paper suggests ways to load site specific symmetric keys onto a PIV Card after the card has been issued, which allows each smart card to share a unique secret key with each PACS with which it interacts. The paper presents four protocols that enable a Card Management System (CMS) to securely load site specific PACS symmetric keys. Each protocol presents unique security characteristics and uses the PIV Card's card management key in different capacities.

Disclaimer

This is research paper intended to stimulate discussion on new concepts and approaches. Statements made are the opinions of the authors and should not be interpreted as standards, guidelines, best practices, or recommendations for specific changes to any other NIST publications.

Keywords: HSPD-12, Key Management, PIV, FIPS 201, Smart Cards, PACS, Card Authentication Key

Table of Contents

1. Introduction

In many facilities, a primary consideration in the selection of an authentication mechanism for a Physical Access Control system (PACS) is the need for speed. Individuals attempting to enter the facility need to be able to quickly swipe their smart cards near the proximity reader and be granted access if they are authorized. A common form of authentication in these systems involves simply reading a serial number from the card, which is then checked against an access control list. This is a very weak form of authentication, however, since it would be very easy for an unauthorized person to create a smart card that responds with the same serial number as that of a smart card belonging to an authorized individual.

The use of a cryptographic challenge/response mechanism as part of the authentication process can minimize the risk that a card can be successfully cloned. Where a smart card only needs to support access to a single facility, a secret key that is shared between the smart card and the facility's PACS may be loaded onto the smart card during the initial card personalization process.

Homeland Security Presidential Directive 12 [2] (HSPD-12) requires the issuance of smart cards Personal Identity Verification Cards (PIV Cards) to federal employees and contractors, with the intention that these smart cards be interoperable across federal agencies. That is, an employee of one agency should be able to use a PIV Card issued by his or her agency to gain entry into any federal building to which he or she is authorized. The use of a single secret key that is loaded onto the PIV Card at the time of personalization is not appropriate in this scenario, as it would require the secret keys of individuals' PIV Cards to be widely shared among a large number of independently operated PACS.

Challenge/response protocols that are based on public key cryptography are well-suited for use in a multi-organizational environment. Every PIV Card includes an asymmetric authentication key and certificate. Use of a challenge/response protocol that employs the asymmetric authentication key would allow for anyone to authenticate any PIV Card without the need to share secret keys. In some environments, however, the amount of time required for a smart card to perform a private key operation may be considered unacceptable.

In this paper, we describe architecture for securely injecting secret keys onto smart cards. In this architecture, each smart card shares a unique secret key with each PACS with which it interacts. When an individual wishes to be able to access a facility controlled by a PACS for which no key has yet been loaded onto the card, a protocol is followed in which the smart card, the PACS, and the CMS that initialized the smart card work together to load a new secret key onto the card. The resulting architecture allows for the use of a fast, secure authentication mechanism based on secret key cryptography in a multi-organizational environment while maintaining a minimal sharing of secret keys.

2. Overview

Ideally, the process of loading a symmetric key onto a smart card for the purpose of enabling authentication for physical access would only involve two entities, the PACS and the smart card. When a card holder presents a smart card to a door reader in order to obtain access to a facility, the PACS needs to be able to authenticate the card, but the card does not need to authenticate the PACS. For this reason, while a PACS needs to ensure that it does not provide keys that will be accepted for authentication purposes to cards that it has not authenticated, it is not strictly necessary for a smart card to authenticate the source of a symmetric key before accepting the key for storage on the card. In practice, however, smart cards have limited memory, and it cannot be assumed that a smart card will have sufficient storage space to hold every symmetric key that is provided to it. Thus, there is a practical need to manage the limited storage space available.

Making meaningful decisions about which keys to store on the card will require, at a minimum, the ability to authenticate the sources of the keys, a process that is too complicated for a smart card to perform by itself. For every smart card, however, there is a CMS that is responsible for the personalization of that card. The protocols presented in this paper include the smart card's CMS, along with the smart card and the PACS, in the key loading process. The CMS is responsible for managing the smart card's resources. The CMS decides whether a key presented by a PACS should be stored on the card, and if the card's memory is already full, which PACS's key should be deleted in order to free up space for the new key.

Federal Information Processing Standards 201 (FIPS 201) [7] specifies one mandatory key and four optional keys for inclusion on PIV Cards. One limitation of the FIPS 201 specification is that it does not specify any asymmetric keys that are under the control of the PIV Card rather than the card holder. The card management key is under the control of the PIV Card, but it is a symmetric key. This limits the use of the card management key to secure communication between the two entities that know the key: the PIV Card and the PIV Card's CMS. The remaining keys defined in FIPS 201 are under the control of the card holder. This means that data encrypted using the key management public key may be read by the card holder and data signed using the PIV authentication key, the card authentication key, or the digital signature key may have been created by the card holder.

All four of the protocols in this paper rely on the card management key to authenticate messages sent between the PIV Card and the PIV Card's CMS. In all four protocols this key is used to allow the PIV Card to authenticate the key injection command. In the third and fourth protocols, this key is also used to allow the PACS to authenticate the PIV Card, with the aid of the PIV Card's CMS. The second protocol, on the other hand, makes use of a yet to be defined asymmetric key management key to allow the PACS to send encrypted data to the PIV Card.

Other protocols to symmetric key management are possible, although we find the motivating arguments less compelling. A fixed set of PACS keys could be loaded by the CMS into the PIV Card at issuance (could not adapt to a changing population of PACS, or a universe of PACS keys larger than could be stored on the PIV Card). After authentication using the asymmetric PIV authentication key or biometric mechanism, a PACS could load a new PACS key into a key cache on the PIV Card (would require the cache to be public-writable, and cache management could be difficult on smart cards without real-time clocks). A PIV Card and PACS could generate a new shared session key through execution of a key establishment protocol between them (would produce unique, independent keys for each PIV Card-PACS pair, complicating reader access to PACS keys). The approaches that offer the best security and scalability rely on the CMS as a trusted intermediary for symmetric key injection onto the PIV Card.

3. A Simple Key Injection Protocol

This section presents the simplest of the four protocols described in this paper. Unlike the other three protocols, this protocol makes no attempt to prevent the PIV Card's CMS from learning the value of the symmetric physical access key provided by the PACS to the PIV Card.

1. The card holder connects to the PACS over a two-way authenticated Transport Layer Security (TLS) session [6] in which client authentication is performed using the PIV Authentication key from the PIV Card.
2. The PACS uses information provided by the card holder in addition to information available from the Backend Attribute Exchange [1] to determine whether to provision the PIV Card with a symmetric physical access key. The PACS also uses the Backend Attribute Exchange to find the Universal Resource Identifier (URI) for the PIV Card's CMS.
3. If the PACS wishes to provision the PIV Card with a symmetric physical access key then the PACS directs the card holder to the PIV Card's CMS by providing the card holder with a URI that points to the card holder's CMS. The URI includes an identifier for the PACS (e.g., https://cms.example.com/keyloader?siteid=192.0.2.4).
4. The card holder follows the URI, which results in the establishment of a two-way authenticated TLS session with the PIV Card's CMS. Client authentication is performed using the PIV Authentication key from the PIV Card.
5. The CMS makes a decision as to whether to load the PACS's key onto the PIV Card, and proceeds with the remainder of the protocol only if the key is to be loaded. The decision process must include verifying that the identifier specified in the URI corresponds to a legitimate PACS. If the identified PACS is legitimate, but no empty slots remain on the PIV Card, then the CMS may ask the card holder to determine which, if any, PACS's key should be deleted in order to allow the new key to be loaded.
6. The CMS uses the Backend Attribute Exchange to obtain the symmetric physical access key for the PIV Card, K_{PHYS}, from the PACS. The communication between the CMS and the PACS is performed over a two-way authenticated TLS session using a cipher suite that provides both confidentiality and integrity protection for the data transmitted between the CMS and the PACS. Before providing K_{PHYS}, PACS verifies that the CMS requesting the key is the CMS responsible for the PIV Card.
7. The CMS determines in which slot, S_{PIV}, the symmetric key should be stored on the PIV Card.
8. The CMS generates a key injection command with the following data: C_s, K_{PHYS}, I_p, S_{PIV}. C_s is a counter that is shared between the CMS and the card that is incremented for each key injection command and that is used in order to allow the PIV Card to detect message replays. I_p is the PACS's site identifier. The key injection command is encrypted and protected using a Message Authentication Code (MAC). The encryption and MAC keys are derived from the symmetric card management key, K_{CM}.
9. The CMS sends the key injection command to the PIV Card.
10. The PIV Card decrypts and verifies the authenticity of the key injection command, checks that the counter, C_s, is higher than for any previously processed key injection command, and then stores K_{PHYS} and I_p in slot S_{PIV}. The PIV Card then sends a response to the CMS indicating that the key injection command has been successfully executed.

4. A Card Key Management Certificate

FIPS 201 specifies an optional card authentication key, which may be either symmetric or asymmetric, that is under the control of any entity that has physical possession of the PIV Card. The card authentication key is intended to be used for authentication to PACS in a manner that provides for strong single factor authentication. In this section, we discuss the potential benefits of replacing the card authentication key with a card key management key that is under the control of the PIV Card. As will be described, this card key management key could be used to support single factor authentication, two factor authentication, and symmetric key injection.

The infrastructure requirements for the card key management key would be similar to the requirements for the asymmetric card authentication key. The key would have to be generated on the PIV Card and the PIV Card could not permit exportation of the private key from the card. The certificates for card key management public keys would look just like the certificates issued for card authentication public keys with the exception that the keyUsage extension would assert the keyEncipherment or keyAgreement bit instead of the digitalSignature bit. In addition, the card key management private key could be used without explicit user action. The main difference would be that the card key management private key could not be used directly via a card command. Instead, the card key management private key could only be used by applets loaded onto the PIV Card, and these applets would not be permitted to export any data (keys) encrypted using the card key management public key. This restriction is necessary in order to protect against multi-protocol attacks, in which an attacker replays a message in the context of one protocol that was originally created for use in another protocol. For example, if an attacker could simply send a command to the card to decrypt some data using the card key management key, then the attacker could eavesdrop on a symmetric key injection protocol exchange and then send a command to the card asking it to decrypt the encrypted symmetric key that it captured while eavesdropping.

Appendix D of NIST Interagency Report 7452 (NISTIR 7452) [5] presents a protocol for establishing a secure session between a PIV Card and a smart card reader. The protocol relies on the use of a key management private key that is under the control of the PIV Card and whose corresponding public key appears in a certificate that can be validated by the card reader. The proposed card key management key could be used for this purpose. The result of the protocol is a pair of symmetric keys, one for encryption and the other for authentication, that are shared between the PIV Card and the card reader.

Single factor authentication could be achieved by running the key establishment portion of the protocol in Appendix D of NISTIR 7452, using the card key management key, and then having the card reader perform a challenge/response using one of the symmetric keys created during the session establishment protocol. Two factor authentication could be achieved by performing secure biometric match-on-card as described in NISTIR 7452.

In addition to being able to replace the card authentication key and to being able to support protocols such as secure biometric match-on-card [5], the presence of card key management keys on PIV Cards would also be ideal for supporting the injection of symmetric keys onto PIV Cards. A protocol for using a card key management key to perform symmetric key injection is presented below. This protocol differs from the protocol in Section 3 in that it ensures that thesymmetric key remains a secret shared by only the PACS and the PIV Card. Neither the PIV Card's CMS nor the PIV Card holder may obtain a copy of the symmetric key.

4.1 Symmetric Key Injection Using a Card Key Management Key

1. The card holder connects to the PACS over a two-way authenticated TLS session in which client authentication is performed using the PIV Authentication key from the PIV Card.
2. The PACS uses information provided by the card holder in addition to information available from the Backend Attribute Exchange to determine whether to provision the PIV Card with a symmetric physical access key. The PACS also uses the Backend Attribute Exchange to find the URI for the PIV Card's CMS.
3. If the PACS wishes to provision the PIV Card with a symmetric physical access key then the PACS directs the card holder to the PIV Card's CMS by providing the card holder with a URI that points to the card holder's CMS. The URI includes an identifier for the PACS (e.g., https://cms.example.com/keyloader?siteid=192.0.2.4).
4. The card holder follows the URI, which results in the establishment of a two-way authenticated TLS session with the PIV Card's CMS. Client authentication is performed using the PIV Authentication key from the PIV Card.
5. The CMS makes a decision as to whether to load the PACS's key onto the PIV Card, and proceeds with the remainder of the protocol only if the key is to be loaded. The decision process must include verifying that the identifier specified in the URI corresponds to a legitimate PACS. If the identified PACS is legitimate, but no empty slots remain on the PIV Card, then the CMS may ask the card holder to determine which, if any, PACS's key should be deleted in order to allow the new key to be loaded.
6. The CMS uses the Backend Attribute Exchange to obtain the symmetric physical access key for the PIV Card from the PACS. The communication between the CMS and the PACS is performed over a two-way authenticated TLS session. The PACS performs the following steps to respond to the CMS's request for a key:
    a. Use the Backend Attribute Exchange to obtain the PIV Card's card key management certificate.
    b. Validate the PIV Card's card key management certificate and verify that the entity identified by the Federal Agency Smart Credential Number (FASC-N) [8] in the subjectAltName extension in the certificate should be provided with a symmetric PACS key.
    c. Encrypt the symmetric PACS key, K_{PHYS}, that is to be provided to the PIV Card using the public key from the PIV Card's card key management certificate: $(K_{PHYS})K_{CKMK}$.
    d. Send $(K_{PHYS})K_{CKMK}$ to the CMS as the value for the requested attribute.
7. The CMS determines in which slot, S_{PIV}, the symmetric key should be stored on the PIV Card.
8. The CMS generates a key injection command with the following data: C_s, $(K_{PHYS})K_{CKMK}$, I_p, S_{PIV}. C_s is a counter that is shared between the CMS and the card that is incremented for each key injection command and that is used in order to allow the PIV Card to detect message replays. I_p is the PACS's site identifier. The key injection command is protected using a MAC. The MAC key is derived from the symmetric card management key, K_{CM}.
9. The CMS sends the key injection command to the PIV Card.

10. The PIV Card verifies the authenticity of the key injection command, checks that the counter, C_s, is higher than for any previously processed key injection command, uses its card key management private key to decrypt $(K_{PHYS})K_{CKMK}$, and then stores K_{PHYS} and I_p in slot S_{PIV}. The PIV Card then sends a response to the CMS indicating that the key injection command has been successfully executed.

5. Symmetric Key Injection Without a Card Key Management Key

This section presents a protocol that does not rely on the presence of a card key management key that results in a symmetric key that is only shared between the PACS and the PIV Card. In this variation of the protocol, the PIV Card establishes a connection with the PACS. This allows the PACS to authenticate the PIV Card using the card's PIV authentication key. In the case that the PIV Card connects to the PACS from a client that is trusted by the PACS, this allows the PACS to verify that the symmetric key that it creates can only be obtained by the PIV Card. In the case that the PIV Card is connected to the PACS from a client that is not trusted by the PACS, this allows the PACS to verify that the PIV Card is present, but the PACS must rely on the PIV Card's CMS to ensure that no entity other than the PIV Card can obtain the symmetric key.

5.1 The Protocol

1. The card holder connects to the PACS over a two-way authenticated TLS session in which client authentication is performed using the PIV Authentication key from the PIV Card.
2. The PACS uses information provided by the card holder in addition to information available from the Backend Attribute Exchange to determine whether to provision the PIV Card with a symmetric physical access key. The PACS also uses the Backend Attribute Exchange to find the URI for the PIV Card's CMS.
3. The PACS sends its Rivest Shamir Adleman (RSA) key management key, $K_{KM,PACS}$ to the PIV Card.[1]
4. The PIV Card generates a random session key, K_s, and a nonce, N_s.[2]
5. The PIV Card sends the following data to the PACS:
    a. encrypted session key and nonce: $(K_s \mid\mid N_s)K_{KM,PACS}$.
    b. $\mathrm{MAC}_{KPACS} = \mathrm{MAC}[K_{CM}, (K_s \mid\mid N_s)K_{KM,PACS} \mid\mid K_{KM,PACS}]$, where K_{CM} is the symmetric card management key that the PIV Card shares with its CMS.
6. The PACS directs the card holder to the PIV Card's CMS by providing the card holder with a URI that points to the card holder's CMS. The URI includes an identifier for the PACS, the encrypted session key and nonce and the MAC (e.g., https://cms.example.com/keyloader?siteid=192.0.2.4&keyandnonce=...&macKPACS=...).
7. The card holder follows the URI, which results in the establishment of a two-way authenticated TLS session with the PIV Card's CMS. Client authentication is performed using the PIV Authentication key from the PIV Card.
8. The CMS makes a decision as to whether to load the PACS's key onto the PIV Card, and proceeds with the remainder of the protocol only if the key is to be loaded. The decision process must include verifying that the identifier specified in the URI corresponds to a legitimate PACS. If the identified PACS is legitimate, but no empty slots remain on the PIV Card, then the CMS may ask the card holder to determine which, if any, PACS's key should be deleted in order to allow the new key to be loaded.
9. The CMS verifies that K_s and N_s were created by the PIV Card and that the PIV Card used the correct key to encrypt K_s and N_s:

[1] Note that this is just a raw RSA key, not a certificate.

[2] The PIV Card only needs to store K_s and N_s until the key injection operation completes. K_s must only be shared between the PIV Card and the PACS.

   a. The CMS uses the PACS's identifier from the URI to obtain the PACS's key management certificate and then validates the PACS's key management certificate.

   b. The CMS extracts the public key, $K'_{KM,PACS}$, from the key management certificate.

   c. The CMS extracts $(K_s \| N_s)K_{KM,PACS}$ and MAC_{KPACS} from the URI and verifies that $\text{MAC}[K_{CM}, (K_s \| N_s)K_{KM,PACS} \| K'_{KM,PACS}] = \text{MAC}_{KPACS}$.

10. The CMS establishes a two-way authenticated TLS session with the PACS and sends the PIV Card's FASC-N and $(K_s \| N_s)K_{KM,PACS}$ to the PACS.

11. The PACS:

    a. Verifies that $(K_s \| N_s)K_{KM,PACS}$ is the same as the encrypted session key and nonce that it received directly from the PIV Card.

    b. Generates the symmetric PACS key, K_{PHYS}, for PIV Card.

    c. Encrypts K_{PHYS} using the session key generated by the PIV Card: $(K_{PHYS})K_s$.

    d. Submits N_s and $(K_{PHYS})K_s$ to the PIV Card's CMS over the TLS protected channel.

12. The CMS performs the following steps:

    a. Determine in which slot, S_{PIV}, the symmetric key should be stored on the PIV Card.

    b. Generate a key injection command with the following data: N_s, $(K_{PHYS})K_s$, I_p, S_{PIV}. I_p is the PACS's site identifier. The command is protected using a MAC computed using K_{CM}.

13. The CMS sends the key injection command to the PIV Card.

14. PIV Card verifies authenticity of the key injection command, checks that the command includes the correct nonce, N_s, uses K_s to decrypt $(K_{PHYS})K_s$, and then stores K_{PHYS} and I_p in slot S_{PIV}. The PIV Card then sends a response to the CMS indicating that the key injection command has been successfully executed.

5.2 An Analysis of the Protocol

The first step in the protocol involves the PACS authenticating the smart card. After the card's identity has been authenticated, that PACS may use the authenticated identity to determine whether it wishes to load a symmetric physical access key onto the card. Note that the authentication performed in the first step of the protocol does not ensure the PACS that the session key sent to it in step four was created by the smart card (unless the card holder connects to the PACS using a client that is trusted by the PACS). While step one ensures that the smart card is present, the session key may have been generated by the card holder on a desktop computer. The PACS must rely on the smart card's CMS's verification of MAC_{KPACS} to ensure that the session key was generated by the smart card.

Once the smart card has been authenticated, the PACS sends its asymmetric key management public key to the card. Since the card is not expected to authenticate the PACS, the PACS only needs to send its public key, rather than a certificate that contains the key. The smart card responds by creating a fresh session key and nonce, which it encrypts using the PACS's public key and sends back to the PACS. The PACS will later encrypt the symmetric physical access key using the session key, as a way of ensuring that only the smart card can obtain the key. The nonce will be included in the key injection command so that the smart card can verify that the command is not a replay and that the key to be loaded was actually encrypted using the session key that is currently in memory.

The smart card also computes a MAC that covers both the encrypted session key and nonce and the public key used to encrypt these values. This MAC will be sent to the smart card's CMS for verification. The CMS's verification of the MAC will protect against two attacks. First, since the key used to compute the MAC, K_{CM}, is under the control of the smart card and not the card holder, verification of MAC_{KPACS} ensures that the session key was created by the smart card and not by a separate device, such as a desktop computer, that is under the control of the card holder. Second, when combined with the validation of the PACS's key management certificate and the verification that the key management key used by the smart card to encrypt the session key is the same as the public key in the PACS's key management certificate, the CMS helps to protect against man-in-the-middle attacks.

6. An Alternative Protocol for Symmetric Key Injection Without a Card Key Management Key

This section presents an alternative key injection protocol, which differs from the protocol presented in Section 5 in that the PIV Card does not connect to the PACS prior to connecting with its CMS. The result of this difference is that the PACS cannot authenticate the PIV Card at all and so must completely rely on the PIV Card's CMS to ensure that symmetric key that it provides can only be obtained by the PIV Card.

6.1 The Protocol

1. The card holder connects to the CMS over a two-way authenticated TLS session in which client authentication is performed using the PIV Authentication key from the PIV Card.
2. The PIV Card's CMS establishes a two-way authenticated TLS session with the PACS.
3. The CMS obtains the PACS's RSA key management certificate, validates it, and then sends the public key from the certificate, $K_{KM,PACS}$, to the PIV Card.
4. PIV Card generates a random session key, K_s, and a nonce, N_s.[3]
5. PIV Card sends the following data to the CMS:
    a. encrypted session key and nonce: $(K_s \parallel N_s)K_{KM,PACS}$.
    b. $MAC_{KPACS} = MAC[K_{CM}, (K_s \parallel N_s)K_{KM,PACS} \parallel K_{KM,PACS}]$, where K_{CM} is the symmetric card management key that the PIV Card shares with its CMS.
6. The CMS verifies that K_s and N_s were created by the PIV Card and that the PIV Card used the correct key to encrypt K_s and N_s by verifying that $MAC[K_{CM}, (K_s \parallel N_s)K_{KM,PACS} \parallel K_{KM,PACS}] = MAC_{KPACS}$.
7. The CMS sends the following data to the PACS over the TLS protected channel:
    a. The FASC-N from the PIV Card's validated PIV Authentication certificate.
    b. $(K_s \parallel N_s)K_{KM,PACS}$
8. The PACS uses the information provided by the CMS to determine whether to provision the PIV Card with a key. If the PACS decides to provision the PIV Card with a key, then the PACS performs the following steps:
    a. Generate the symmetric PACS key, K_{PHYS}, for PIV Card.
    b. Encrypt K_{PHYS} using the session key generated by the PIV Card: $(K_{PHYS})K_s$.
    c. Submit [FASC-N, N_s, $(K_{PHYS})K_s$, I_p], where I_p is the PACS's site identifier, to the PIV Card's CMS over the TLS protected channel.
9. The CMS performs the following steps:
    a. Determine in which slot, S_{PIV}, the symmetric key should be stored on the PIV Card.
    b. Generate a key injection command with the following data: N_s, $(K_{PHYS})K_s$, I_p, S_{PIV}. The command is protected using a MAC computed using K_{CM}.

[3] The PIV Card only needs to store K_s and N_s until the key injection operation completes. K_s must only be shared between the PIV Card and the PACS.

10. PIV Card verifies authenticity of the key injection command, checks that the command includes the correct nonce, N_s, uses K_s to decrypt $(K_{PHYS})K_s$, and then stores K_{PHYS} and I_p in slot S_{PIV}.

6.2 An Analysis of the Protocol

While this protocol is similar in many ways to the protocol in Section 5, the security properties are very different. Since there is no direct communication between the PACS and the PIV Card, there is no way for the PACS to verify that K_s was generated by the PIV Card, or even to verify that the PIV Card is present. All of the information provided to the PACS by the PIV Card's CMS could have been created by the CMS without the participation of the PIV Card. For this reason, the PACS must place a greater degree of trust in the CMS than would be required if the PIV Card were to communicate directly with the PACS, as in the protocol in Section 5. The main advantage of the protocol proposed in this section over the protocol in Section 3 is that the steps performed by the PIV Card are exactly the same as in the protocol in Section 5. Thus, the PIV Card could support both protocols using the same code.

7. Implementation Considerations

7.1 Site Identifiers

The site identifiers that are loaded onto the PIV Cards along with the symmetric keys are used by the door readers to indicate to the PIV Cards which key should be used to perform an authentication. A door reader sends a site identifier to the card, the card scans through its list of stored keys for the one with the corresponding site identifier, and uses that key to perform the challenge. Thus, site identifiers must be selected in a manner that satisfies two requirements: a smart card must not be required to store two different keys at the same time with the same site identifier and door readers must be able to determine at the time that a challenge command is sent to a card what site identifier to specify in the command.

In many PACS, there is only one-way communication between door readers and door controllers. Door readers collect information from the smart card and forward that information to door controllers, which make access control decisions. In such environments, door readers cannot look up smart card specific site identifiers to include in challenge messages. For this reason, it is recommended that each PACS select a single site identifier for use with every smart card to which it provides a key. In order to ensure that no smart card will receive keys from two different PACS with the same site identifier, each PACS must ensure that its site identifier is unique. Rather than creating a new system for assigning site identifiers to PACS, this paper recommends the use of IPv4 addresses, which would result in site identifiers being 32 bits in length. As every PACS will need to be network accessible, every PACS will have already been assigned an IPv4 address, which it can use as its site identifier.

7.2 Generating Symmetric PACS Keys

There are three options for generating symmetric PACS keys for injection onto smart cards: a PACS may generate a unique, randomly generated key for each card, load the same key onto every card, or load each card with a unique key that is derived from a common master secret key. In this paper, we recommend the use of derived keys for most PACS.

When using derived keys the PACS maintains a single key derivation key, which it distributes to all of the door controllers. Keys for each PIV Card are created deterministically from the key derivation key and the PIV Card’s FASC-N using a key derivation function [4]. While the individual PIV Card’s key may be easily computed using the key derivation key and the card’s FASC-N, it is infeasible for an attacker who learns the value of one or more of the derived keys to determine the value of the key derivation key or the value of any other derived key. Thus, the compromise of a PACS key on one or more PIV Cards would not require the replacement of the key derivation key, only the removal of the compromised PIV Cards from the access control list. The use of derived keys also avoids the need for each door controller to maintain a separate key for each PIV Card. A door reader may send a challenge to a PIV Card and then send the challenge, the PIV’s response, and the PIV Card’s asserted FASC-N to the door controller. The door controller can authenticate the PIV Card using only the key derivation key and the information received from the door reader. The door controller may then make an authorization decision based on the authenticated FASC-N.

In practice, each door controller will need to maintain several key derivation keys, since a key derivation key may only be used for a limited period of time [3]. Since PIV Cards may be valid for up to five years, if key derivation keys are changed every year then at any time a door controller will need to maintain five key derivations keys. When attempting to authenticate a PIV Card, the door controller would need to attempt to verify the response from the PIV Card using each of the five key derivations keys, and would consider the PIV Card to be successfully authenticated if response could be verified using any of the key derivation keys.

7.3 Service Architecture

The architecture supporting symmetric key injection needs to support two basic scenarios: loading the key of an individual PACS onto an existing PIV Card and the bulk loading of keys onto a new PIV Card.

The protocols presented in this paper are all designed to allow the card holder to connect to the PACS and/or CMS using an Internet connected client. This allows a card holder to prepare in advance for a visit to a site by loading that site's key onto his/her PIV Card prior to arrival. In the case of a conference, for example, the conference registration web site could direct holders of PIV Cards to the site's PACS. Card holders who arrive at a site who do not have an appropriate key loaded onto their card could be directed to a kiosk running a web browser that allows access to the PACS's web site and to the web site's of known CMS's.

When a card holder obtains a new PIV Card, the symmetric PACS keys from the old PIV Card cannot be copied onto the new card. However, since the CMS knows the identities of all of the PACS that have keys stored on the old card, the CMS may contact each of these PACS in order to obtain a key for the new card. By performing this bulk loading of symmetric keys onto the new card, the disruption to the card holder resulting from obtaining a new PIV Card is minimized. In the protocols in Sections 3 and 4, the PIV Card's initial contact with the PACS serves only to obtain the PACS's identifier, and so these initial steps may be skipped during the bulk loading process. The initial steps in the protocol in Section 5, on the other hand, cannot be skipped. Thus, if neither the protocol from Section 3 nor the protocol from Section 4 is being used, then either the protocol from Section 6 should be used for all key loading operations or the protocol from Section 6 should be used for bulk key loading onto new PIV Cards and the protocol from Section 5 should be used for loading individual keys onto existing cards.

No matter which protocol is used for loading keys onto new PIV Cards, some method should be provided to enable PACS to link the FASC-N from a new PIV Card to the FASC-N from an old PIV Card. This information may be used by the PACS to aid in the decision as to whether to provide a key for the new PIV Card and also to enable the PACS to update their access control databases with the new identifier. The linkage between the old and new FASC-N could be included in the protocol messages or could be included as an attribute in the Backend Attribute Exchange.

8. Use of Symmetric Key for Authentication

Once symmetric physical access keys have been loaded onto smart cards, they may be used to support one, two, or three factor authentication. One factor authentication may be performed using the same protocol as is specified for the symmetric card authentication key in FIPS 201. Two factor authentication may be performed by requiring the use of either a PIN or a biometric in addition to requiring the use of the symmetric key in order to prove the presence of the PIV Card. Three factor authentication may be performed by requiring the use of a PIN and a biometric in addition to requiring the use of the symmetric key in order to prove the presence of the PIV Card.

While FIPS 201 specifies that secret key operations may be performed using the card authentication key without explicit user action, this does not need to be the case for symmetric keys that are injected onto PIV Cards. The protocols in this paper indicate that when a key is loaded onto a PIV Card, a site identifier, I_p, is associated with the key. A second piece of information, a boolean indicating whether operations may be performed using the key without explicit user action may also be associated with the key. If this flag is set then the PIV Card cannot successfully respond to a challenge unless the correct PIN is provided to unlock the key. If a key is designated as requiring a PIN to unlock, then that key should not be accessible over the contactless interface. If the key were accessible over the contactless interface then an attacker could lock someone's PIV Card by sending a sequence of PIN attempts to the card without the card holder's knowledge.

Biometric authentication could be achieved by using the symmetric key to perform secure biometric match-on-card [5], which would result in either two or three factor authentication depending on whether the symmetric key was configured to require PIN entry before use. In the case that PIN entry is required; the biometric template may be transferred from the PIV Card to the door reader, allowing the door reader to perform the match rather than the PIV Card.

Appendix A—References

[1] Backend attribute exchange architecture and interface specification, May 2008. http://www.smart.gov/awg/documents/BackendArchitectureInterfaceSpec.pdf.

[2] HSPD 12. *Policy for a Common Identification Standard for Federal Employees and Contractors*, August 2004.

[3] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. *Recommendation for key management - part 1: General. NIST special publication 800-57,* National Institute of Standards and Technology, March 2007.

[4] Lily Chen. *Recommendation for key derivation using pseudorandom functions.* NIST special publication 800-108, National Institute of Standards and Technology, November 2008.

[5] David Cooper, Hung Dang, Philip Lee, William MacGregor, and Ketan Mehta. *Secure biometric match-on-card feasibility report.* NIST Interagency Report 7452, November 2007.

[6] Tim Dierks and Eric Rescorla. *The Transport Layer Security (TLS) Protocol Version 1.2.* RFC 5246 (Proposed Standard), August 2008.

[7] FIPS 201-1. *Personal Identity Verification (PIV) of Federal Employees and Contractors*. National Institute of Standards and Technology, March 2006.

[8] PACS v2.2. *Technical Implementation Guidance: Smart Card Enabled Physical Access Control Systems*, Version 2.2. The Government Smart Card Interagency Advisory Board's Physical Security Interagency Interoperability Working Group, July 2004.