Authenticating Mandatory Access Controls and Preserving Privacy for a High-Assurance Smart Card

Helmut Scherzer[1], Ran Canetti[2], Paul A. Karger[2], Hugo Krawczyk[2,3], Tal Rabin[2], and David C. Toll[2]

[1] IBM Deutschland GmbH, Secure Systems and Smart Cards, Schönaicher Str. 220, D-71032 Böblingen, Germany
scherzer@de.ibm.com

[2] IBM Research Division, T. J. Watson Research Center, PO Box 704, Yorktown Heights, NY 10598, USA
{canetti,talr,toll}@us.ibm.com, {karger,hugo}@watson.ibm.com

[3] Department of Electrical Engineering, Technion, Haifa, 32000, Israel
hugo@ee.technion.ac.il

Abstract. This paper presents an authentication protocol for high-assurance smart card operating systems that support download of mutually suspicious applications. Such a protocol is required to be part of the operating system, rather than the traditional smart card approach of allowing applications to do authentication, because strong authentication is essential for the operating system to protect one application from another. The protocol itself is based on the existing IKE protocol [13], used for authentication in IPSEC. What is new is the integration of an IKE-like protocol with authentication of mandatory secrecy and integrity access controls, the recognition that a single PKI-hierarchy cannot certify identity and all possible mandatory access rights, and the use of IKE to resolve privacy problems found in existing smart card authentication protocols.

1 Caernarvon – A High-Assurance Smart Card OS

IBM® has been developing a secure smart card operating system, called Caernarvon, that is intended to be evaluated at the highest assurance levels of the Common Criteria [17]. The Caernarvon system is written for, and requires, processors containing hardware security features that provide separate supervisor and user modes, and hardware memory protection. The security of the system is then enforced by the hardware, rather than being entirely dependent on software. The initial version is being written for the Philips Smart*XA2* processor, which is a 16-bit smart card processor containing the required hardware security features.

The operating system is designed to permit applications developers to solve security-related problems that could never be addressed before, either in smart cards or in larger computer systems. In particular, the security model and the operating system are designed to permit in-the-field downloading of applications written either in native languages (such as C or assembler) or in interpreted languages (such as Java Card$^{\mathrm{TM}}$). These downloaded applications could be mutually hostile, yet the operating system will prevent unauthorized interference between the applications, yet still allow controlled sharing of information between selected applications, subject to the constraints of the new security model. The applications themselves do not need to be evaluated or certified. The only obvious exception to this is the special *guard* applications that are privileged to change the security markings of files.

The Caernarvon system contains a Mandatory Security Policy; this consists of a modified Bell and LaPadula [3] lattice secrecy model, together with a modified Biba [4] integrity model. Space does not permit discussion of the Caernarvon Mandatory Security Policy in detail, but policy is described in more detail in [22,23,24,29]. Girard [12] has also suggested the use of mandatory security policies in smart cards.

2 High Assurance and Authentication

One of the fundamental requirements for a high assurance operating system is authentication. This authentication must be performed by the operating system itself, not by an application, so that the operating system both is guaranteed and can guarantee to others that the authentication has been completed. The operating system then knows, with high assurance, the identity of the *user*, in this case the outside world, namely the system behind the smart card reader. The system can use this knowledge to safely grant the user access to files and other system objects; conversely, it can also use this knowledge to ensure, with high assurance, that a user is denied access to anything he is not authorized to see or use.

If the operating system were to follow the current smart card practice of delegating all authentication to the application, then true protection of applications from one another is impossible. For example, if authentication were delegated to a poorly written vending machine application, it might claim mistakenly to have properly authenticated the card issuer's primary administrator. Without strong authentication, such as that provided by the Caernarvon operating system, the vending machine application could mistakenly grant access to the card issuer's administrative files stored on the card.

Furthermore, even operating systems that have been successfully evaluated and certified to the highest levels of the Common Criteria might not be perfect. Authentication provides the first line of defense in any operating system, and should an attacker get past an application-provided authentication scheme, even if only improperly authenticated as a legitimate user of that application, then that attacker has gained additional access to software running on the card that could then be targeted. While a successful high-assurance evaluation gives a very high level of confidence that attack is not possible, there will always remain the small possibility of a vulnerability that escaped detection. The U.S. National Computer Security Center recognized this possibility in its so-called, "Yellow Book" [31] that provided guidance on when to use operating systems that have been evaluated to different levels. The "Yellow Book" recognized that even an A1-evaluated operating system, the highest level possible under the "Orange Book" [10], was not sufficiently strong to protect the very highest levels of compartmented intelligence information against totally uncleared users.

3 Authenticating Multi-organizational Mandatory Access Controls

Caernarvon is designed to support mandatory access controls in a commercial setting. This means that the lattice security models must contain entries from multiple organizations. This is very different from most military applications of mandatory access controls where the access controls have a common definition. It is beyond the scope of this paper to discuss the implications of mandatory access controls from different organizations. An early discussion of some of these issues can be found in [20], and a preliminary design for multi-organizational mandatory access controls can be found in [21], although this design has already undergone a number of changes. For purposes of this paper, it is sufficient to say that the smart card must be able to handle mandatory access controls from multiple organizations.

Both the Bell and LaPadula secrecy model [3] and the Biba integrity model [4] provide a lattice structure of non-hierarchic access classes. Each object in the system is assigned an access class, and each user is assigned a security clearance that is also an access class. Access control decisions are made by comparing the access class of an object with the access class of the referencing user or process. The details of access classes are unimportant to this paper. What is important is that both the security and integrity lattices may contain access classes from different mutually suspicious organizations, and that possession of these access classes must be authenticated. Note that this type of multi-organizational access class is much more general than the access classes typically used in the US Department of Defense, such as those defined in FIPS PUB 188 [30] or the DoD Common Security Label [8].

Proving that either a server or a card actually holds a particular organizational access class is not easy. The conventional approach of a Public Key Infrastructure (PKI) with a trusted third-party Certification Authority (CA) is not likely to be acceptable, because of the extreme sensitivity of mandatory access classes in some applications. For example, despite the fact that the NATO countries are all close allies, there is not likely to be a single CA that all of the NATO countries would accept to prove that someone holds a particular national security clearance.

Instead, a organizational Security Authority (SA) is defined, which serves as the agency to digitally sign certificates that prove that a particular organizational access class is held. An SA is very similar to a CA, except that there is no hierarchical tree above the various SAs. Each SA stands alone.

Therefore, both servers and smart cards must hold digital signatures from multiple organizations' SAs and must be prepared to verify those digital signatures. Each SA with an access class on the card or on a server must sign a hash of that organization's access class, cryptographically bound to the public key of the card or server.

IBM has developed a protocol for defining access classes and SAs on a smart car. However, that protocol is beyond the scope and page limitations of this paper, and will be the subject of a future paper. This paper will simply assume that SA public keys for the appropriate organizations are available on the card.

The mandatory security policy requires that an effective authentication of its users (in this case, the outside world) be performed before allowing access to objects under the control of the operating system. The security policy in particular, applies to application download, application selection, and inter-application information sharing and exchange.

The authentication scheme must be application independent, and enforced by the Caernarvon kernel. The authentication protocol as described here has been submitted to the ESIGN-K [2] working group that is designing a specification for the European signature application on smart cards and to the Global Platform organization (http://www.globalplatform.org) that is developing standards for multi-application smart cards based on earlier work by Visa.

The Caernarvon authentication combines four mechanisms:

– a device verifies the existence of a certified secret key on the other party.
– the devices negotiate or exchange information to establish a common session key for subsequent operations.
– the devices negotiate or exchange information to establish a common access class for subsequent operations.
– the authentication is mutual, that is it is two-way, and binds all of the above elements.

The session encryption uses a symmetric algorithm, for performance reasons. Therefore this document describes the derivation of symmetric keys, and does not consider an option for asymmetric keys. Once the session key is established, a trusted channel is available to protect or conceal the information transmitted over the interface. The application of Secure Messaging (SM) is mandatory for subsequent operations to ensure the provision of a trusted channel.

There are current smart card standards for digital signature applications [6,7] that use a two-way authenticated Diffie-Hellman key agreement scheme in order to create a triple-DES session key for the current session. This key agreement scheme is described in ISO/IEC 11770-3 [19], section 6.7 "Key agreement mechanism 7"; this in turn is based on the three-pass authentication mechanism of ISO/IEC 9798-3 [18]. However, we show in section 5 below, that these existing smart card standards have privacy problems. Therefore, Caernarvon uses a Diffie-Hellman key agreement scheme based on the SIGMA design [25], standardized by the Internet Key Exchange Protocol (IKE) [13]. This was chosen so that authentication for Caernarvon could be based on existing standards, and because the security and privacy properties of SIGMA protocols have been formally proven in [5].

The cryptographic principles behind Caernarvon authentication are not new. What is new is the application of IKE to smart cards to resolve the privacy problems in existing smart card standards and the combination of IKE authentication with multi-organizational mandatory access controls in a high-assurance operating system. The Caernarvon protocol also solves a potential problem of a man-in-the-middle modifying the Diffie-Hellman public parameters. In normal usage, IKE does not suffer from this problem, but it can arise in other contexts. See section 6 for details.

4 Diffie-Hellman Key Exchange

Diffie-Hellman was the first public-key algorithm openly published in 1976 [9]. The Diffie-Hellman algorithm was first developed by M. J. Williamson at the Communications-Electronics Security Group (CESG) in the UK and published internally somewhat later in [33], but that work remained classified until much later [11]. It gets its security from the difficulty of calculating discrete logarithms in a finite field, as compared with the ease of performing exponentiation calculations in the same field.

4.1 Simple Diffie-Hellman Key Exchange, No Authentication

The general form of an unauthenticated Diffie-Hellman key exchange is as follows: in this, A is the card reader and B is the Caernarvon smart card.

Both A (the card reader) and B (Caernarvon) share the public quantities p, q and g where:

- p is the modulus, a prime number; for security, this number should be of the order of 1024 bits or more.
- q is a prime number in the range of 159-160 bits
- g is a generator of order q, that is $g^q = 1 \bmod p$ and for all i where $i < q$, $g^i \neq 1 \bmod p$

Then the protocol proceeds as follows:

1. A chooses random number a with $1 \leq a \leq q-1$, and B chooses random number b with $1 \leq b \leq q-1$
2. A computes $K_A = g^a \bmod p$ and sends it to B. B computes $K_B = g^b \bmod p$ and sends it to A
3. A computes $K_{AB} = (K_B)^a \bmod p$. B computes $K_{BA} = (K_A)^b \bmod p$.
4. A deletes the random number a. B deletes the random number b.

A and B have derived the same number (the key) because:

$$K_{AB} = (K_B)^a = (g^b)^a = (g^a)^b = (K_A)^b = K_{BA}$$

4.2 Authenticated Diffie Hellman

The algorithm described above in section 4.1 produces an agreed secret key; however, because there is no authentication included, it is vulnerable to man-in-the-middle attacks. ISO/IEC 11770-3 [19] section 6.7 “Key agreement mechanism 7” uses a Diffie-Hellman style of key exchange, and also involves mutual authentication. The following shows the general flow of such a scheme.

Stage 1. The reader, A, calculates K_A as above, and concatenates it with a certificate containing A's identity A and public key PK_A. A transmits this to B:

$$\begin{tikzcd}\mathbf{A} \arrow[r, "Cert(A\|PK_A)_{CA}\|K_A"] & \mathbf{B}\end{tikzcd}$$

Fig. 1. Key Agreement Mechanism 7 of ISO 11770-3, Part 1

Stage 2. The card, B, checks the certificate received from A. B then calculates K_B and K_{BA} as above, and generates a message for A consisting of the following items concatenated together:

1. K_B
2. a certificate containing B's identity B and public key PK_B
3. a signature of the concatenation of A's identity A, K_A and K_B, signed using B's secret key SK_B corresponding to the public key PK_B
   Note that this signature consists of just the signature value - it is not necessary to include the quantities being signed, since A already has them (they were sent to B in Stage 1 above), or are included as part of this message.
4. a cryptographic check, using a keyed hash function **f** as a message authentication code with the key K_{BA}, of A's identity A, K_A and K_B.

B then transmits this to A:

$$\begin{tikzcd}\mathbf{A} & \mathbf{B} \arrow[l, "K_B\|Cert(B\|PK_B)_{CA}\|Sig_{SK_B}(A\|K_A\|K_B)\|\mathbf{f}_{K_{BA}}(A\|K_A\|K_B)"']\end{tikzcd}$$

Fig. 2. Key Agreement Mechanism 7 of ISO 11770-3, Part 2

Stage 3. A now checks the certificate received from B, verifies the signature (using B's public key PK_B contained in the certificate), and calculates K_{AB}, and uses this to verify the cryptographic check using the function **f** and the key K_{AB}.

A then generates a message consisting of the following items concatenated together:

– a signature of the concatenation of K_A, K_B and B's identity B, signed using A's secret key SK_A corresponding to the public key PK_A
  Note that this signature consists of just the signature value - it need not include the quantities being signed, since B already has them.
– a cryptographic check, using the crypto function **f** with the key K_{AB}, of K_A, K_B and B's identity B.

A then sends this to B:

$$\begin{tikzcd}\mathbf{A} \arrow[rrrr, "Sig_{SK_A}(B\|K_A\|K_B)\|\mathbf{f}_{K_{AB}}(B\|K_A\|K_B)"] & & & & \mathbf{B}\end{tikzcd}$$

Fig. 3. Key Agreement Mechanism 7 of ISO 11770-3, Part 3

Stage 4. Finally, B verifies the signature (using A's public key PK_A transmitted in the certification in Stage 1), and verifies the cryptographic check using the function $\mathbf{f}$ and the key K_{BA}. If the two verifications succeed, B signifies his acceptance to A:

$$\begin{tikzcd}\mathbf{A} & & & & \mathbf{B} \arrow[llll, "\text{OK}"']\end{tikzcd}$$

Fig. 4. Key Agreement Mechanism 7 of ISO 11770-3, Part 4

5 Privacy-Preserving Protocol

The protocol based on ISO 11770-3, discussed in section 4.2 above, has the disadvantage that the card ("B" in the discussion) reveals his identity and certificate before he has verified the credentials of the reader "A". This could be viewed as a violation of the privacy of the card holder - the identity and certificate of the card B are revealed, not just to the reader A, but also to anyone eavesdropping on the communications between the reader and the card. The reader A might not be physically co-located with the card, but actually connected via a network of some sort. The DIN standards [6,7] for digital signature cards suffer from this potential privacy problem.

The protocol based on ISO 11770-3, discussed in section 4.2 above, also has the disadvantage that the number of bits transmitted in all the stages is somewhat larger than necessary. Minimizing the total number of bits transmitted is important, because some smart card readers will only communicate at 9600bps, and even ignoring the cost of computing the cryptographic operations, the time needed to transmit all the bits could become a serious problem in response time to the card holder.

To resolve both the privacy problems and to reduce the number of bits to be transmitted, Caernarvon bases its authentication on the SIGMA design [25] and the Internet Key Exchange (IKE) standard [13]. This protocol offers several significant advantages:

1. The session key parameters are exchanged very early in the protocol, even before the authentication has been completed. In this way, the information exchanged in the protocol, including the peers' identities can be protected from third-party eavesdropping.
2. A discloses its identity and credentials to B first; B reveals its identity and credentials only after verifying those of A. This prevents revealing the card holder's identity to a reader that cannot be authenticated or that cannot prove that it is authorized for a particular mandatory access classes. Therefore, the card's identity is protected not only against eavesdropping, but also against an active (man-in-the-middle) attacker. The reader's identity is not protected against an active attacker, but presumably the reader has fewer privacy concerns than the card holder. Note that in all authentication protocols, one party must reveal its identity first, and that party's privacy will always be subject to active attacks of this kind.
3. IKE transmits fewer bits in total. This will improve performance on slow readers.
4. The SIGMA and IKE protocols followed here have been rigorously analyzed and proven correct [5], which is a major benefit in any system planning to be evaluated at the highest levels of the Common Criteria. In particular, see [25] for more details on the cryptographic rationale of these protocols and the subtle cryptographic attacks they prevent.

This section contains a cryptographic description of the authentication protocol used by Caernarvon. Note that in contrast to the protocol described in section 4.2, the Caernarvon protocol starts as in unauthenticated Diffie-Hellman, and then authenticates A before B exposes his identity. The crucial technical difference between these protocols is that in the case of the Caernarvon protocol, A can authenticate itself to B without having to know B's identity, while in the ISO protocol of section 4.2, A authenticates to B by signing B's identity (thus requiring the knowledge of B's identity by A before A can authenticate to B)[1].

As discussed in section 4.1, A (the reader) and B (Caernarvon) share the public quantities p, q, and g. Section 6 will discuss why these public quantities must themselves be authenticated.

Stage 1. A chooses a random number a with $1 \leq a \leq q-1$, computes a key token $K_A = g^a \bmod p$, and transmits it to B.

$$\begin{tikzcd} \mathbf{A} \arrow[rrr, "K_A"] & & & \mathbf{B} \end{tikzcd}$$

Fig. 5. Authentication Stage 1: A sends a key token to B

[1] The Caernarvon protocol described here assumes that only one authentication is in progress between a reader and a smart card at any one time. This follows current practice in the smart card industry. If support for multiple authentication sessions were desired, the protocol would have to be modified to include a cryptographically bound session identifier.

Stage 2. B chooses a random number b with $1 \leq b \leq q-1$, computes a key token $K_B = g^b \bmod p$, and transmits it to A.

$$\begin{tikzcd} \mathbf{A} & & & \mathbf{B} \arrow[lll, "K_B"'] \end{tikzcd}$$

Fig. 6. Authentication Stage 2: B sends a key token to A

At this point, neither A nor B has revealed his identity. However, they now can compute a mutual key K_{AB} as in section 4.1. Using the mutual key K_{AB}, they can derive additional keys K_{ENC}, for encrypting messages and K_{MAC}, for computing message authentication codes (MACs) as specified in section 7.2.

Stage 3. A now sends its certificate to B by encrypting it with K_{ENC}. A now computes E_{01} as shown below:

$$E_{01} = 3DES\_Encrypt_{K_{ENC}}(Cert(A))$$

A now transmits E_{01} together with its MAC to B.

$$\begin{tikzcd} \mathbf{A} \arrow[rrr, "E_{01} \| MAC_{K_{MAC}}(E_{01})"] & & & \mathbf{B} \end{tikzcd}$$

Fig. 7. Authentication Stage 3: A sends certificate to B

Stage 4. B responds with a challenge. From a strictly cryptographic perspective, stage 4 could be combined with stage 2, reducing the total number of message flows. However, this is a protocol for smart cards, and it must fit into the existing standard for smart card commands [15] and use the GET CHALLENGE and EXTERNAL AUTHENTICATE commands.

$$\begin{tikzcd} \mathbf{A} & & & \mathbf{B} \arrow[lll, "\text{RND.B}"'] \end{tikzcd}$$

Fig. 8. Authentication Stage 4: B sends challenge to A

Stage 5. A now computes E_1 as shown below:

$$E_1 = 3DES\_Encrypt_{K_{ENC}}(A \| Sig_{SK_A}[K_A \| A \| RND.B \| K_B \| DH(g \| p \| q)])$$

A now transmits E_1 and a MAC of E_1 to B. The signature is a signature with message recovery, so all parameters in the signature can considered to be recoverable. The Diffie-Hellman key parameters are part of the signature in order to provide authenticity of the parameters. See section 6 for details.

$$\begin{tikzcd}[column sep=huge] \mathbf{A} \arrow[r, "E_1\|MAC_{K_{MAC}}(E_1)"] & \mathbf{B} \end{tikzcd}$$

Fig. 9. Authentication Stage 5: Authenticate A

At the conclusion of stage 5, B has authenticated A. It is at this point that the Caernarvon authentication protocol mandatory access control checks, as described below in section 7.5, steps 11 and 12. The mandatory access checks are done here, so that B can verify mandatory access rights, before revealing any privacy-sensitive information to A.

Stage 6: B now verifies the MAC, decrypts E_1, and verifies the signature using A's public key PK_A. B has now authenticated A and knows that K_A and K_B are fresh and authentic. However at this point, while B knows there is no man-in-the-middle because B checked the signature from A, A does not know who he is talking to, and hence is unsure if there may be a man-in-the-middle attack. B computes E_{02} (its encrypted certificate) and sends it to A.

$$E_{02} = 3DES\_Encrypt_{K_{ENC}}(Cert(B))$$

$$\begin{tikzcd}[column sep=huge] \mathbf{A} \arrow[r, leftarrow, "E_{02}\|MAC_{K_{MAC}}(E_{02})"] & \mathbf{B} \end{tikzcd}$$

Fig. 10. Authentication Stage 6: B sends certificate to A

Stage 7. A sends a challenge to B. Just as for stage 4, strict cryptographic requirements could reduce the total number of message flows. However, once again, it is desirable to use the ISO standard [15] GET CHALLENGE and EXTERNAL AUTHENTICATE commands.

$$\begin{tikzcd}[column sep=huge] \mathbf{A} \arrow[r, "\text{RND.A}"] & \mathbf{B} \end{tikzcd}$$

Fig. 11. Authentication Stage 7: A sends challenge to B

Stage 8. B now computes E_2 as shown below:

$$E_2 = 3DES\_Encrypt_{K_{ENC}}(B\|Sig_{SK_B}[K_B\|B\|RND.A\|K_A])$$

The signature is a signature with message recovery, so all parameters in the signature can considered to be recoverable.

B now transmits E_2 and a MAC of the value E_2 to A.

$$\begin{tikzcd}\mathbf{A} & & \mathbf{B} \arrow[ll, "E_2 \| MAC_{K_{MAC}}(E_2)"']\end{tikzcd}$$

Fig. 12. Authentication Stage 8: authenticate B

A can now verify the MAC and decrypt E_2. Using the chain of certificates back to the root CA, A can verify the certificate from the IC manufacturer for B, which contains B's identify B and public key PK_B. Thus A knows, and can trust, B's public key PK_B. Hence A can now authenticate B by verification of the signature:

$$Sig_{SK_A}[K_A \| A \| RND.B \| K_B \| DH(g \| p \| q)]$$

6 The Authenticity of the Public DH Parameters

The proof given in [5] assumes that the public DH parameters are authentically known. This assumption does not necessarily hold for signature cards, as the public DH parameters may be retrieved from a file on the card prior to the device authentication.

As a consequence, the authenticity of these public key parameters is not given implicitly. Therefore the public key parameters need to be signed by the reader, which allows the card to verify that the reader used its correct parameters.

The following scenario demonstrates how this weakness could be used for an attack, given that the public DH parameters were not signed by the reader.

This attack scenario was contributed to the E-Sign committee by Andreas Wiemers [32]. Related attacks have been described by Lim and Lee [27] and Antipa, et. al. [1].

Step	**READER(A)**		**CARD (B)**
1		$\longleftarrow$	p, q, g
2	$K_A = g^a \bmod p$	$\longrightarrow$	
3		$\longleftarrow$	$K_B = g^b \bmod p$
4			$K_{AB} = {K_A}^b \bmod p$
5		$\longleftarrow$	RND.B
6	$\tilde{K}_{AB} = {K_B}^a \bmod \tilde{p}$		
7	$SIG_A(K_A \| RND.B \| K_B)$	$\longrightarrow$	
8	$RND.A$	$\longrightarrow$	
9		$\longleftarrow$	$SIG_B(K_B \| RND.A \| K_A)$

As soon as K_{AB} is available it is used to secure the communication between the reader and the card (after session key generation).

We suggest that an attacker changes the transmission such, that he replaces p by $\tilde{p}$. We obtain:

Step	READER(A)		CARD (B)
1		$\longleftarrow$	$\tilde{p}, q, g$
2	$\tilde{K}_A = g^a \bmod \tilde{p}$	$\longrightarrow$	
3		$\longleftarrow$	$K_B = g^b \bmod p$
4			$\tilde{K}_{AB} = \tilde{K}_A^b \bmod p$
5		$\longleftarrow$	RND.B
6	$\hat{K}_{AB} = {K_B}^a \bmod \tilde{p}$		

In general $\hat{K}_{AB}$ and $\tilde{K}_{AB}$ are different. However, if the attacker chooses $\tilde{p} = c \cdot p$ with a small number c (e.g. $c = 2$), an attacker observes the case $K_{AB} = 1 \bmod c$ with an expected probability of approximately $\frac{1}{c}$. In this case, it is easy to prove that the equation $\tilde{K}_{AB} = \hat{K}_{AB} = K_{AB}$ is valid exactly if $K_{AB} \equiv 1 \bmod c$.

If the reader and the card may communicate with secure messaging without error indication, the attacker obtains information about the negotiated key. For example, if c is 2, then the attacker would know that K_{AB} is odd, thus leaking one bit of the key. Choosing c larger makes it less probable to hit a valid negotiation between the reader and the card. However if this hit occurs, more bits are leaked from K_{AB}. As K_A and K_B are computed from a and b, which are chosen randomly, the attack cannot be used to accumulate information about the negotiated keys. However in the sense of provable security this is a valid example to demonstrate the general observation that the authenticity of public parameters is important for proper security.

Reason: For $\tilde{p} = c \cdot p$, it is always valid

$$\tilde{K}_{AB} = (g^a \bmod pc)^{r_B} \bmod p = g^{a \cdot b} \bmod p = K_{AB}$$

As $0 \leq K_{AB} \leq c \cdot p$ and $0 \leq K_{AB} < p$, then $\hat{K}_{AB} = K_{AB}$ is true if this equality is mod p and mod c. As the equality mod p is trivial and $\hat{K}_{AB} = {K_B}^a \bmod c \equiv 1$, the assumption is confirmed.

7 Caernarvon Authentication Flow

This section describes the full flow of the authentication protocol used by the Caernarvon kernel, including implementation details omitted from the cryptographic description above in section 5 and the authentication of mandatory access controls of section 3. It will also include smart-card specific details. Each stage from section 5 will be discussed in a separate sub-section. Each stage is implemented as one or more steps. While the numbering of sub-sections, stages, and steps is complex, it helps cross-referencing back to the pure cryptographic protocol of section 5 and ultimately to the implementation source code (not included in this paper.)

The reader is authenticated first; this is to provide the highest achievable protection of the ICC's (i.e. the card's) identity and data until the IFD (the reader) has been authenticated.

7.1 Stage 1 – Reader Sends Key Token to the Card

The five steps that make up stage one are explained below.

Step 1. Power the card
When the smart card is first inserted into the reader, it receives power and must respond following the Answer to Reset protocol specified in [14].

Step 2. Read Cryptographic Token Information
Prior to authentication the IFD might require parameters on how to proceed with, and where to find resources relevant for, the authentication. The IFD may issue commands to retrieve information for the authentication parameters in a format specified by ISO 7816-15 [16]. In particular, it may retrieve information on the authentication algorithm, the format of certificates, the presence of specific certificates, and key related information (key ID's, key length etc.)

Step 3. Read key exchange parameters
If the reader does not have the required key exchange information available, it may read them from the card. These may include public algorithm quantities which depend on the authentication algorithm. For instance, in a Diffie Hellman Key exchange scheme the public key quantities would be the public parameters p, q and g. The public key quantities reveal information about the authentication mechanism. As long as these quantities are used in many cards, the identity of a card is not revealed by the leakage of this information prior to authentication of the reader.

Step 4. Reader Computes its Key Token and Sends it to the Card
The reader chooses a random number a, and computes its key token $K_A = g^a \bmod p$ (see section 4). It then sends the key token to the card.

7.2 Stage 2 – Card Sends Key Token to the Reader

The two steps that make up stage two are explained below.

Step 5. Card Computes its Key Token and Sends it to the Reader
Caernarvon chooses a random number b, and computes its key token $K_B = g^b \bmod p$. It then sends the key token to the reader.

Step 6. Derive Keys for Use During the Remainder of Authentication
At this point, the reader and the card have completed simple unauthenticated Diffie-Hellman key agreement, as described in section 4.1. Neither side has been authenticated yet, but using the common secret K_{AB}, they now derive an encryption key K_{ENC} and a MACing key K_{MAC} that will

be used to protect the remainder of the authentication protocol from casual eavesdroppers. Both keys are 112-bit 3DES keys. Key derivation from the common secret is according to ANSI X9.63 [28] using HMAC [26] as the pseudo-random function. Note that there could still be a man-in-the-middle at this point in the protocol.

Both the reader and the card calculate:

$HASH1 = HMAC[K_{AB}](1)$
and
$HASH2 = HMAC[K_{AB}](HASH1\|2)$

112 bits are selected from $HASH1$ to produce K_{ENC}, and 112 bits are selected from $HASH2$ to produce K_{MAC}.

7.3 Stage 3 – Sending the Reader's Certificate

The two steps that make up stage three are explained below:

Step 7. Selection of the CA Public Verification Key
This step may be a single stage, or two stages, depending on whether the CA public key is held in the card. If the key is present in the card, then the reader need only select the key. If the key is not present in the card then the reader has to send the CA's public key with the appropriate certificate which must be verified by the card before it can be used. Care must be used in implementing this step, because a malicious reader could attempt a denial of service or a privacy attack against the card. If the card stores the CA's public key and certificate on a long-term basis, the reader could run the card out of memory, simply by sending lots of public keys and certificates. Since authentication has not yet been completed, the card cannot use memory quotas to protect itself, because it does not yet know whose memory quota to charge. The same approach could be used to violate privacy by using the public keys and certificates as the equivalent of web browser cookies. Fortunately, there is an easy fix. The card simply needs to not store these intermediate keys and certificates after either authentication has completed or the card has been reset. That simple fix eliminates the possibility of either denial of service or privacy attacks.

Step 8. Verify Reader's Certificate
The reader sends a computer verifiable (CV) certificate containing its public key. The IFD executes the VERIFY CERTIFICATE command, which causes Caernarvon to verify the certificate using the public key of the certification authority selected or verified in step 5. The public key of the reader cannot be trusted until it is confirmed by an appropriate certificate. The signature and the use of HMAC ensure that the operation is fresh and is cryptographically tied to the values of K_A and K_B.

7.4 Stage 4 – Card Sends a Challenge

Step 9. Get Challenge:

In order to prove its authenticity dynamically, the reader requests a challenge from the ICC. The challenge consists of a simple random number. Cryptologically, K_A and K_B have sufficient random quality. The additional request for $RND.B$ is used to initialize certain cryptographic checksum counters used in the secure messaging protocol of [14].

7.5 Stage 5 – Authenticate the Reader

Step 10. External authentication:

The reader computes a signature on the concatenation of K_A, A's identity, the challenge, K_B, and the public Diffie-Hellman parameters. Including the public parameters avoids the attack described in section 6.

After Step 10, the reader's public key can be trusted, and the card knows that there is no man-in-the-middle.

Step 11. Choose Secrecy Access Class

This step is to verify the secrecy access class to be requested, which may be less than the maximum secrecy access class of either the reader or the card. Note that this step is entirely optional; if it is omitted then a secrecy level of System Low will be used by default. This procedure is performed at this point in the authentication because:

- the session key has been agreed, so this negotiation can be performed using encrypted transmissions, to prevent leakage of information as to what access authorizations permitted to each participant. Thus this step cannot be performed earlier in the authentication.
- it is completed now rather than later for privacy reasons, that is so that the card does not reveal any personal information until it has been verified that the reader is indeed authorized to see such information. Revealing the card's identity before authenticating the reader provides for the possibility of unauthorized tracking the movements of the card holder.

Initially, the reader (IFD) must prove its secrecy access class to the card. The data field passed is an access class, signed and cryptographically tied to the public key of the IFD. Caernarvon verifies the IFD's access to the IFD (by verification of the appropriate signatures from the SA and that the public keys match), and then, in the response message, returns proof that the card is authorized to use the same access class. This proof consists of the card's access class signed by the SA and cryptographically tied to the card's public key.

There is no privacy problem revealing this information, because at this point it has been verified that the IFD holds the proper access classes. The IFD cannot complete its verification of the card's access classes until after step 16 below, because the IFD cannot be sure of the card's public key and the absence of a man-in-the-middle attack until step 16 has been completed. However, Caernarvon returns the access classes here because the card does know that it is safe to do so, and it avoids the necessity of implementing an additional step and APDU in the sequence after step 16.

At this point, the card now knows the maximal access class that can be used. The IFD must next specify exactly what access class (less than or equal to the maximal) that is to be used for this session. It is possible to reduce this access class, for example if the access class contains several categories, to select a subset of these categories, or to select a lower secrecy level than permitted by the verified access class. (In principle, the IFD could authenticate at precisely the access class that it wished to use. However, this would require that the secrecy authority for the access class to have signed all possible combinations of access classes less than or equal to the maximal access class. Storing all those signatures would be quite impractical, particularly when the IFD's memory capacity may be as limited as the smart card itself.) This selection of the subset could be done at a later stage in the authentication; however, it seems sensible to do it at the same time as the verification.

Step 12. Choose Integrity Access Class

This step is to verify the integrity access class to be requested. This procedure is performed at this point in the authentication for the same reasons as for the secrecy access class. Note that this step is entirely optional; if it is omitted then an integrity level of System Low will be used by default. Initially, the reader (IFD) must now prove its integrity access class to the card. The data field passed is signed and cryptographically tied to the public key of the IFD. This has now set a maximal access class (i.e. integrity level) that can be used. It is possible to reduce this access class, to select a lower integrity level than permitted by the verified access class. This selection of the subset could be done at a later stage in the authentication; however, it seems sensible to do it at the same time as the verification.

7.6 Stage 6 – Sending the Certificate to the Reader

Step 13. Read Card's Certificate

The reader now needs to read the certificate of the card. In the E-Sign protocol, this certificate may be signed by an external CA and there may be a need to chain back through one or more certificates. However, for a high-assurance system, simply chaining back through certificates to some public CA does not give the reader any assurance that the smart card is actually running the genuine high-assurance Caernarvon operating system. Instead, the public keys of each Caernarvon card are signed by the card manufacturer who is responsible for ensuring that the correct high-assurance ROM image has been burned into the card and that the card has been properly and securely initialized. No other authority can provide better assurance, because ultimately it is the card manufacturer who controls the ROM image.

Step 14. Read Card Manufacturer's Certificate

If the reader does not already have the card manufacturer's certificate, it reads in from the card.

Step 15. Key selection

Before the reader can process the INTERNAL AUTHENTICATE command, the card's private authentication key must be selected, using a Manage Security Environment command.

7.7 Stage 7 – Reader Requests Card to Send Challenge to Reader

Step 16. Internal Authentication

The reader issues an INTERNAL AUTHENTICATE command. The causes the card to send its challenge to the reader. The card then computes the signature over the challenge and the key token K_A, K_B and returns it to the reader encrypted with secure messaging.

7.8 Stage 8 – Reader Authenticates Card

Step 17. Verifying the signatures

The reader now verifies the response with the trusted key from the card's certificate and gets evidence that the signer (holder of the certificate) is identical with the entity that made the key negotiation.

After Step 17, both sides are authenticated and mandatory access classes have been selected and verified.

7.9 Post-authentication Phase

A successful authentication selects the desired access class, negotiated from that provided in the certificate of the IFD, for the current session. The session keys for the session are available to both parties, as described in section 7.2. All further communication is done under Secure Messaging either through protection (MAC) or encryption of the data being transmitted at the interface. The decision to send subsequent blocks in encrypted form depends on the selected secrecy access class. In general, secrecy access classes higher than certain specified values will require session encryption. The session ends when the card is RESET or powered off. There is no way to start a fresh authentication without RESETting the card.

8 Conclusion

We have shown how a high-assurance smart card operating system that supports download of mutually suspicious applications must enforce its own high security authentication protocol, rather than allowing the traditional smart card approach of allowing individual applications do perform their own authentication. Strong operating system-based authentication is essential so that the operating system can reliably protect one application from another, yet still permit controlled sharing of information. The protocol is designed to support mandatory access controls for both secrecy and for integrity. We have also shown potential privacy problems with existing smart card authentication protocols and how our new protocol helps to preserve the privacy of the smart card holder. However, the protocol is based on existing authentication standards that have been formally proven, and is being submitted for possible standardization.

Acknowledgements

We acknowledge the useful comments and suggestions of Jonathan Edwards, Rosario Gennaro, and Elaine Palmer of IBM, and Helmut Kurth of atsec information security. Andreas Wiemers of the BSI provided assistance in writing section 6. We also thank Philips Semiconductors for their continuing support. Finally, we thank the anonymous reviewers of the program committee and Yves Deswarte for their suggested improvements to the paper.

References

1. Adrian Antipa, Daniel Brown, Alfred Menezes, René Struik, and Scott Vanstone. Validation of elliptic curve public keys. In Yvo G. Desmedt, editor, *Public Key Cryptography – PKC 2003*, volume 2567, pages 211–223, Miami, FL, 2003. Springer Verlag.
2. Application interface for smartcards used as secure signature creation devices: Part 1 - basic requirements. Technical Report CEN/ISSS WS/E-Sign Draft CWA Group K Version 1.05, Secretariat: DIN Deutsches Institut für Normung e.V, Berlin, 7 May 2003.
3. David E. Bell and Leonard J. LaPadula. Computer security model: Unified exposition and multics interpretation. Technical Report ESD–TR–75–306, The MITRE Corporation, Bedford, MA, USA, HQ Electronic Systems Division, Hanscom AFB, MA, USA, June 1975. `http://csrc.nist.gov/publications/history/bell76.pdf`.
4. Kenneth J. Biba. Integrity considerations for secure computer systems. Technical Report ESD–TR–76–372, The MITRE Corporation, Bedford, MA, USA, HQ Electronic Systems Division, Hanscom AFB, MA, USA, April 1977.
5. Ran Canetti and Hugo Krawczyk. Security analysis of IKE's signature-based key-exchange protocol. In Moti Yung, editor, *Advances in Cryptology - Crypto 2002*, volume 2045 of *Lecture Notes in Computer Science*, pages 143–161, Santa Barbara, CA, 2002. Springer-Verlag.
6. Chipcards with digital signature application/function according to SigG and SigV - part 1: Application interface. Technical Report DIN V66291-1, Secretariat: DIN Deutsches Institut für Normung e.V, Berlin, 15 December 1998.
7. Chipcards with digital signature application/function according to SigG and SigV - part 4: Basic security services. Technical Report DIN V66291-4, Secretariat: DIN Deutsches Institut für Normung e.V, Berlin, 17 October 2000.
8. Common security label (CSL). Technical Report MIL-STD-2045-48501, Joint Interoperability and Engineering Organization (JIEO), Fort Monmouth, NJ, 25 January 1995.
9. W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

10. DOD 5200.28-STD, Department of Defense, Washington, DC, USA. *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985. `http://csrc.nist.gov/publications/history/dod85.pdf`.
11. J. H. Ellis. The story of non-secret encryption. Technical report, Communications-Electronics Security Group (CESG), Cheltenham, UK, 1987. `http://www.cesg.gov.uk/publications/media/nsecret/ellis.pdf`.
12. Pierre Girard. Which security policy for multiapplication smart cards? In *Proceedings of the USENIX Workshop on Smartcard Technology*, pages 21–28, Chicago, IL, 1999. The USENIX Association.
13. D. Harkins and D. Carrel. The internet key exchange (IKE). Technical Report RFC2409, November 1998. `ftp://ftp.rfc-editor.org/in-notes/rfc2409.txt`.
14. Information technology - identification cards - integrated circuit(s) cards with contacts - part 3: Electronic signals and transmission protocols. Technical Report ISO/IEC 7816-3:1997(E), International Organization for Standardization, Genève, 18 September 1997.
15. Information technology - identification cards - integrated circuit(s) cards with contacts - part 4: Inter-industry commands for interchange. Technical Report ISO/IEC 7816-4, International Standards Organization, Genève, 1995.
16. Information technology - identification cards - integrated circuit(s) cards with contacts - part 15: Cryptographic information application. Technical Report ISO/IEC CD 7816-15, draft edition, International Organization for Standardization, Genève, 2001.
17. Information technology - security techniques – evaluation criteria for it security – parts 1, 2, and 3. Technical Report ISO/IEC 15408-1, -2, and -3, International Organization for Standardization, Genève, 1999.
18. Information technology - security techniques - entity authentication - part 3: Mechanisms using digital signature techniques. Technical Report ISO/IEC 9798-3, International Organization for Standardization, Genève, 15 October 1998.
19. Information technology - security techniques - key management - part 3: Mechanisms using asymetric techniques. Technical Report ISO/IEC 11770-3, International Organization for Standardization, Genève, 1 November 1999.
20. Paul A. Karger. The lattice security model in a public computing network. In *ACM 78: Proceedings 1978 Annual Conference*, volume 1, pages 453–459, Washington, DC, USA, 4–6 December 1978. Association for Computing Machinery.
21. Paul A. Karger. Multi-organizational mandatory access controls for commercial applications. Technical Report RC 21673 (97655), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, 22 February 2000. `http://domino.watson.ibm.com/library/CyberDig.nsf/home`.
22. Paul A. Karger, Vernon R. Austel, and David C. Toll. A new mandatory security policy combining secrecy and integrity. Technical Report RC 21717 (97406), IBM Research Division, Thomas J. Watson Research Center, Yorktown Heights, NY, 15 March 2000. `http://domino.watson.ibm.com/library/CyberDig.nsf/home`.
23. Paul A. Karger, Vernon R. Austel, and David C. Toll. Using a mandatory secrecy and integrity policy on smart cards and mobile devices. In *EUROSMART Security Conference*, pages 134–148, Marseilles, France, 13–15 June 2000.
24. Paul A. Karger, Vernon R. Austel, and David C. Toll. Using mandatory secrecy and integrity for business to business applications on mobile devices. In *Workshop on Innovations in Strong Access Control*, Naval Postgraduate School, Monterey, CA, 25-27 September 2000. published on CD-ROM. `http://www.acsac.org/sac-tac/wisac00/wed0830.karger.pdf`.

25. Hugo Krawczyk. SIGMA: the 'SIGn-and-MAc' approach to authenticated diffie-hellman and its use in the IKE protocols. In D. Boneh, editor, *Advances in Cryptology – CRYPTO 2003 Proceesings*, volume 2729 of *Lecture Notes in Computer Science*, pages 399–424, Santa Barbara, CA, 17-21 August 2003. Springer–Verlag.
26. Hugo Krawczyk, M. Bellare, and Ran Canetti. HMAC: keyed-hashing for message authentication. Technical Report RFC-2104, February 1997. `http://www.faqs.org/ftp/rfc/rfc2104.txt`.
27. Chae Hoon Lim and Pil Joong Lee. A key recovery attack on discrete log-based schemes using a prime order subgroup. In Burton S. Kaliski, editor, *Advances in Cryptology - CRYPTO '97*, volume 1294 of *Lecture Notes in Computer Science*, pages 249–263, Santa Barbara, CA, 1997. Springer Verlag.
28. Public key cryptography for the financial services industry, key agreement and key transport using elliptic curve cryptography. Technical Report X9.63-2001, American National Standards Institute (ANSI), 2001.
29. Gerhard Schellhorn, Wolfgang Reif, Axel Schairer, Paul Karger, Vernon Austel, and David Toll. Verification of a formal security model for multiapplicative smart cards. In F. Cuppens, Y. Deswarte, D. Gollmann, and M. Waidner, editors, *6th European Symposium on Research in Computer Security (ESORICS 2000)*, volume 1895 of *Lecture Notes in Computer Science*, pages 17–36, Toulouse, France, 2000. Springer-Verlag.
30. Standard security label for information transfer. Technical Report FIPS PUB 188, National Institute of Standards and Technology, Gaithersburg, MD, 6 September 1994.
31. Technical rationale behind CSC-STD-003-85: Computer security requirements – guidance for applying the department of defense trusted computer system evaluation criteria in specific environments. Technical Report CSC-STD-004-85, DoD Computer Security Center, Fort George G. Meade, MD, 25 June 1985.
32. Andreas Wiemers. Kommentare zu application interface for smart cards used as secure signature creation devices, part 1 - basic requirements version 0.14 28th february 2003 (in German). Technical report, Bundesamt für Sicherheit in der Informationstechnik (BSI), Bonn, Germany, 14 March 2003.
33. M. J. Williamson. Thoughts on cheaper non-secret encryption. Technical report, Communications-Electronics Security Group (CESG), Cheltenham, UK, 10 August 1976. `http://www.cesg.gov.uk/publications/media/nsecret/cheapnse.pdf`.