Electronic Passports in a Nutshell

Wojciech Mostowski and Erik Poll

Radboud University, Nijmegen, The Netherlands
{woj,erikpoll}@cs.ru.nl

Abstract. This document tries to give concise, (semi)formal specifications for the second generation electronic passports as used by most EU countries, and for the closely related ISO18013 standard for electronic driving licenses. We developed these specifications as a follow-up to making open source Java Card implementations of these standards.
Our aim is to provide useful information – implicit in the official specification, but crucial for the overall security – in a simple format that could be useful to anyone implementing these standards, performing security tests, or doing code reviews. More generally, we want to explore useful formats for rigorously specifying the typical complex combinations of security protocols that arise in real applications.
In particular, we provide state diagrams which describe the state that are largely implicit in the official specifications, but which have to be explicit in any implementation, and which also provide a basis for systematic model-based testing.

1 Introduction

Electronic passports, or e-passports for short, contain a contactless smartcard that stores digitally signed data and implements several security protocols to control access to this data. The international standard for e-passports has been specified by the International Civil Aviation Organization (ICAO), in Doc 9303 [5]. The EU has decided to implement additional security checks in second generation e-passports, as specified by the German BSI [1]. These protocols, their shortcomings, or shortcomings in their particular implementations, are discussed in for instance [4, 8, 11, 3].

More recently an ISO standard, ISO18013, has been developed for electronic driving licenses. This standard is very similar to the ICAO and BSI standards for e-passports. The differences are mainly in naming, but also there are minor functional modifications. Since we consider the differences marginal, we will only discuss specifics of the ISO18013 standard in Section 7 towards the end of the paper and concentrate on e-passports in the rest.

The e-passport protocols are sufficiently complex to make understanding them, and correctly implementing them, a non-trivial task. The standards in fact give detailed descriptions of several protocols with some possible variations (e.g. using either RSA or Elliptic Curve Cryptography). In the end every passport implements a particular configuration of a protocol. Understanding the combination and interaction of the various protocols is not so easy. This was our experience when we added Extended Access Control (EAC) functionality to the open source Java Card implementation for the e-passport[1] and when we developed an open source implementation for the ISO18013 e-driving license[2]. For terminal software, which has to interact with smartcards implementing any version of the standards, it is even more complicated.

[1] Available from `http://jmrtd.org`
[2] Available from `http://isodl.sourceforge.net`

To better understand the standards, we developed state diagram models to specify the combined protocols, which we present in this document in sections 4 and 5. For completeness, we also provide Message Sequence Diagrams for all the protocols in Section 6 in the standard abstract security-protocol notation.

Essential differences of these state diagrams with the specifications as given in the official standards are that (i) these models consider the full combination of protocols that have to be implemented, and (ii) the models are *state-based*, i.e. they make explicit which different states any implementation will have to distinguish.

We believe that these models are a useful addition to the official standards. They facilitate good understanding of the protocols and their interaction, and should be useful when developing implementations, in suggesting testing scenarios, and for performing code reviews. Indeed, the models helped to uncover some bugs in our own implementations[3]. We also used the models for model-based testing [9]; this allowed easy experimentation to discover which implementation choices (allowed by the underspecification in the official specs) were made in a particular e-passport implementation.

It would not surprise us if similar diagrams as presented here have been drawn on whiteboards in many places, or appear in (confidential) design documents, test plans or security reviews of e-passports. In fact, anyone implementing the standards will effectively construct a state diagram; given the way a smartcard works, any implementation has to explicitly keep track of state information. We therefore hope that the specifications given here can be of use to others. In fact, it would be nice if official specification documents produced by ICAO, ISO, or BSI would include similar information, even if only as informative rather than normative appendices.[4]

While we try to abstract from the very low-level workings of the particular e-passport protocols in this paper, we also include a detailed, but still lightweight, descriptions of the internals of all e-passport protocols for informative purposes.

AA	Active Authentication
APDU	Application Protocol Data Unit
BAC	Basic Access Control (e-passports)
BAP	Basic Access Protection (ISO18013)
CA	Chip Authentication
DG	Datagroup (a.k.a. a file)
EAC	Extended Access Control (e-passports)
EAP	Extended Access Protection (ISO18013)
MAC	Message Authentication Code
PA	Passive Authentication
RFID	Radio Frequency Identification
SM	Secure Messaging
TA	Terminal Authentication

Table 1. List of abbreviations used in the paper

[3] To be precise, the possibility to send commands unprotected by Secure Messaging after Secure Messaging have been established.

[4] Currently, state diagrams seem to be used only in the supplement to the official specification [6, page 19] for the behaviour of the terminal software.

2 Electronic Identity Documents

An e-passport is a particular case of an electronic identity document. Typically such an e-id document is implemented on a smartcard, either contact (commonly called chip card) or contactless (an RFID tag). Smartcards communicate with the outside world with Application Protocol Data Units (APDUs). APDUs are standardised in the ISO7618 specifications, and for contactless cards the ISO14443 specification defines the lower level contactless protocol. Information requests are sent to the card by an inspection terminal (or, in general, any software with access to a suitable card reader) in so-called command APDUs, and the card provides the requested information and the operation status (so-called status word, which may e.g. indicate insufficient security conditions to complete the request) by sending back a response APDU. Both command and response APDUs are simple, relatively short byte sequences.

One kind of an electronic identity document is one that stores (usually signed and integrity protected) personal data of the holder: names, date of birth, picture, signature, etc., segregated in separate files, in case of e-passports called datagroups. Here e-passports, driving licenses, or national identity cards are notable examples. For instance, on the electronic side, the current Dutch national identity card is practically identical to the Dutch e-passport. A driving license stores very similar data to a passport, but also includes driving specific information, like vehicle categories or limitations. All electronic identity documents in this category are mostly passive, in the sense that they provide read-only data as high level output and internally perform active computations related to access control, protocol integrity checks, and document authenticity control.

Another kind of an electronic identity document are PKI cards used for electronic signatures. Their functionality is of an active nature – their core application is to sign, encrypt, or decrypt input data and return the result. They may or may not store personal data, like name or picture, but they always store private cryptographic keys for internal use and corresponding (state issued) signing certificates.

Obviously a combination of the two categories is possible, or at least a card can be issued with two applications on it, one for passive identity, and one for electronic signatures. More interestingly, some passive identity documents provide enough computing power and functionality to be partly applicable in PKI applications as described in [10]. In particular, the active authentication passport protocol (see below) involves signing of data with a private key stored securely in the passport.

3 e-Passport Protocols

Communication with e-passports involves several dedicated protocols, which run on top of the APDU protocol:

1. *PA (Passive Authentication)*: This is not really a protocol, but refers to the use of digital signatures of data on the passports; for PA the reader has to read several datagroups and check the hashes and digital signatures.
2. *BAC (Basic Access Control)*: The protocol to start communication with the passport chip, in which the reader proves knowledge of the MRZ information physically printed in the passport.
3. *SM (Secure Messaging)*: The protocol to protect the integrity and confidentiality of the communication between the reader and the e-passport. The keys used for SM are established by the BAC protocol, and they are refreshed by the Chip Authentication protocol, see below.
4. *AA (Active Authentication)*: A challenge-response protocol to prove authenticity of the passport chip, in which the e-passport proves knowledge of a private

key for which it has a certificate signed by the issuing country. This protocol is effectively redundant if a passport supports Extended Access Control, as authenticity of the passport chip can then also be established by Chip Authentication, see below.

5. *EAC (Extended Access Control)*: EAC consists of two protocols:
   (a) *CA (Chip Authentication)*: a protocol for the terminal to authenticate the chip. CA is based on a secret key agreement and establishes new SM keys.
   (b) *TA (Terminal Authentication)*: a protocol for the chip to authenticate the terminal, and possibly increase access rights. TA is performed by actively verifying certificates and an authentication request sent to the passport by the inspection system.

   Both these protocols rely on a PKI, where each country issues certificates to its passports (for CA) and certificates to other countries for letting them read the more sensitive data from the passports (for TA). Currently, EAC protects access to the fingerprint and iris scan possibly stored on the passport.

PA, BAC, SM, and AA are specified by ICAO in [5]; only PA is mandatory, the others are optional, but practically all new passports implement BAC, SM, and many passports implement AA. EAC is specified in [1] and it is implemented in all new generation EU passports.

For a passport that supports BAC and EAC, BAC must be performed first, and then EAC is performed by first doing CA and then TA. A typical run would combine the protocols as follows:

$$\text{BAC};\ (\mathsf{select};\ \mathsf{read})^*;\ \text{CA};\ \text{TA};\ (\mathsf{select};\ \mathsf{read})^*$$

where after BAC and after TA the passport reads some datagroups, by first choosing a datagroup with the select command and then reading them with a read command.

At the end of BAC SM is started, and all subsequent protocols are run 'under' SM, which can be schematically illustrated as in Figure 1.

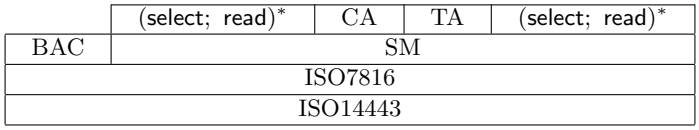


Fig. 1. Protocol stack for the e-passport

Other combinations than those suggested by the regular expression above or the diagram in Figure 1 are possible. For instance: AA could be done after BAC; AA could be done after CA or TA (even though this would be pointless, as CA already authenticates the chip); selecting or reading could happen between CA and TA; and, since TA consists of several steps, AA or reads could even happen during TA, i.e. between individual steps of the TA protocol. The BAC, CA and TA protocols could also be executed in other orders, or possibly multiple times.

The official specifications leave it largely underspecified or implicit if these combinations are allowed and what the response should be. Many of these combinations do not really make sense. Section 2.1.1 of [1] describes the standard e-passport inspection procedure, giving a high-level description of the order in which the various protocols are typically run. Of course, there are no guarantees that any buggy or malicious inspection systems actually stick to normal order. Any secure implementation of the e-passport has to ensure that strange combinations do not lead to

unwanted behaviour. This is why we explore systematic ways of describing wanted – and unwanted – behaviour more precisely in the following sections.

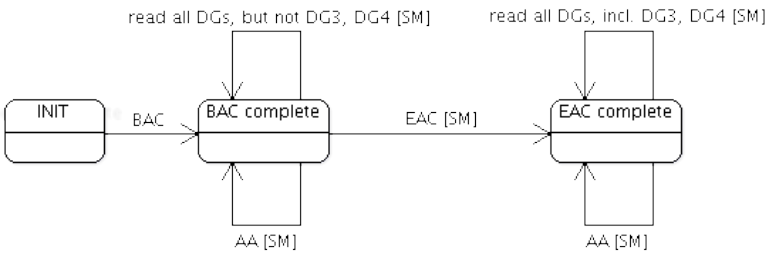
In Section 5 we will come back to the underspecification issues. A more precise description of the passport architecture and internal workings of the protocols is given in Section 6.

4 State Diagrams for the e-Passport

First we sketch the basic issue on how to specify the e-passport by means of a state diagram supplemented with an access control matrix, then we look at a precise and detailed specification.

4.1 Basic State Diagram and Access Control Matrix

On the high level, the e-passport can be in three states, as illustrated by this diagram:



By performing BAC the passport enters a state in which basic datagroups can be read, or AA can be performed. By subsequently performing EAC the passport enters a state in which additional datagroups – e.g. also the fingerprint information – can be read. The SM guard in brackets indicates communication that is encrypted using Secure Messaging.

A diagram like the one above could be supplemented with an access control matrix saying for every state which operations are allowed:

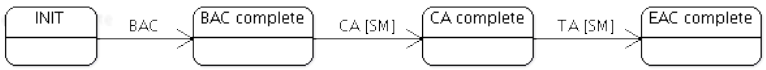
	State		
Operation	INIT	BAC complete	EAC complete
access DG1, DG2	no	yes	yes
access DG3, DG4	no	no	yes
do AA	no	yes	yes
...			

One could also draw the operations in the table as arrows in the diagram, with labels indicating whether the operation should succeed or fail. However, such additional arrows quickly clutter the diagram, especially when more fine grained state set is used as we present below. Also, many of the operations in the access control table will not change the current state, so the diagram would be full of self arrows. Conversely, one could record the diagram arrows in the table, by including the result state from the diagram.

In the following our approach is to use both a diagram to define security status change after different protocol steps and a corresponding access control table indexed by possible operations and diagram states.

4.2 More Detailed State Diagrams

On a lower level of abstraction we distinguish CA and TA as individual steps within EAC:



We can go to lower levels of abstraction still. For instance, BAC consists of two communication steps, which could be distinguished. TA consists of at least 7 steps, and possibly more, depending on the length of the certificate chain that the inspection system provides to the passport.[5]

Figure 2 specifies such a more detailed state diagram for the protocol. Most arrows in the diagram correspond to a single communication from the terminal to the card (with a command APDU) and the resulting response (response APDU) of the card. Only exceptions here are the '(re)set SM key' transitions, which are internal actions of the card we make explicit to keep the diagram more structured (note that the number of the state is unchanged), and the various 'error' transitions, which correspond to the card reporting an error by an appropriate status word in response to a incorrect command APDU sent by the terminal. The diagram includes an (unreachable) 'Personalisation' state, as a reminder that the e-passport must have had additional functionality for initialising the passport. This functionality should of course never be available after the passport has been issued, which is why the state is unreachable.

A passport inspection system can be expected to issue read instructions in state 3 (to read the datagroups protected by BAC), to do AA in state 3 (to check authenticity of the chip), and to read instructions in state 5 (to read the additional datagroups protected by TA). Figure 3 gives the full access control table for the diagram in Figure 2.

5 Underspecification

In the following we discuss some aspects of the passport behaviour that are underspecified in the official documentation.

5.1 Failed Commands

For many errors (e.g. unexpected commands or commands with incorrect information, say an incorrect response to a challenge) it is not clear if the passport should simply ignore them, or abort and go back to the initial state.

For instance, an attempt to read a datagroup in state 2 could be ignored, or lead to a transition to state 1. The same goes for an attempt to read a inaccessible datagroup, say DG3 containing the fingerprint, in state 3. For errors during TA a forgiving implementation might allow a second try at TA (as indicated in Figure 2), but a more restrictive one might abort.

For the overall security it does not seem to matter, if we assume that brute force attacks to break SM or to produce fake certificate chains for TA are not feasible.

Two things are explicitly stated in the specs w.r.t. failed commands. The first is that any error in SM *must* result in aborting the SM session. The second one is that

[5] According to the specification, a single successful certification chain always contains two certificates, but invalid certificates can be presented for verification during the process effectively lengthening the chain of verification operations, we will come back to this later.

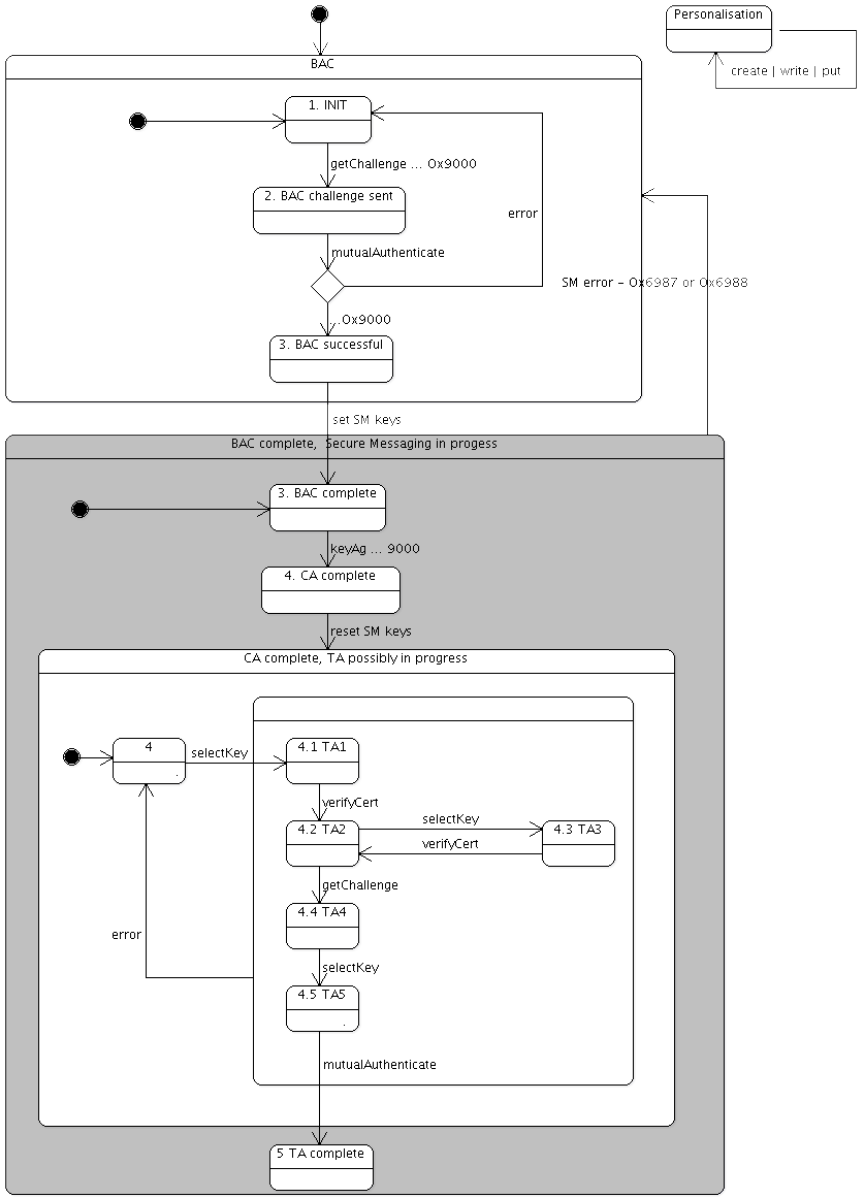


Fig. 2. Detailed state diagram for e-passport implementing BAC and EAC

	pre-BAC	post-BAC	post-CA	post-TA
do BAC, without SM*	yes	no	no	no
read basic DGs	no	yes	yes	yes
read EAC DGs	no	no	no	yes†
do AA	no	yes	yes	yes
do CA	no	yes	no‡	no‡
do TA	no	no	yes	no

* Only getChallenge is not SM protected during BAC. A non-SM getChallenge after BAC could be accepted and treated as the start of a fresh BAC session.
† Only when the terminal certificates gives the right to do this.
‡ Repeated CAs or multiple EACs (TA+CA) could be allowed, but would be pointless.

Fig. 3. Access control matrix, describing (dis)allowed operations at various stages

a single failed certificate verification does not break the whole chain of certificates verified so far during TA, i.e. you can e.g. send one correct certificate and then try 5 different ones if you are not sure which one should be next. Certificate signature is only one of the validity checks performed by the passport on a certificate, the other one is date expiry check. This is why more than one certificate may have to be checked.[6]

5.2 Unexpected getChallenge

An unexpected getChallenge sent without SM, coming at any point, could be treated as the start of a new attempt to do BAC, meaning we go to state 2. A less accommodating implementation would go to state 1.

Similarly, after receiving a getChallenge command in state 2 (which would be second getChallenge in a row) the passport could go to state 1 or stay in state 2.

5.3 Operations Interleaving TA

TA takes several steps – the chain of two certificates has to be verified by the card (selectKey, verifyCert) and a finalising authentication step is performed (getChallenge, selectKey, mutualAuthenticate). A successful TA takes at least 7 APDU commands. It is not clear whether other operations could be interleaved with TA, e.g. reading datagroups, doing AA, or even CA again.

For example, a very strict passport implementation might insist that TA is performed directly after CA. A more liberal one reading datagroups directly after CA, but no longer allow once the first step of TA is taken. An even more liberal one might allow selecting and reading datagroups at any point once BAC has been completed. None of this has any impact on security, assuming of course the access control on reading the datagroups is done correctly. In fact, an EAC passport that we tested allowed such intermediate DG reads.

5.4 Multiple BAC

The specs are not clear about multiple BACs. More specifically: a non-SM getChallenge, at any stage, could be interpreted by the applet as the first step in a new BAC procedure. In this case one can imagine that the passport should bring the read access rights down in case the second (new) BAC procedure fails. Another approach could be simply to ignore such getChallenge and keep the BAC session active.

[6] On the other hand, the passport provides necessary information for the terminal to try only one, most likely to be valid, chain of two certificates.

5.5 Multiple EAC

It is unclear if repeated CAs (CA one after another before continuing with TA) or whole multiple EACs are allowed. There seems no harm in allowing either, moreover, the CA operation always refreshes SM keys, so e.g. the terminal software may wish to perform CA twice to refresh SM keys two times for additional security (regardless of how pointless this is in practice). Even more, one can imagine that EAC is performed twice with two different sets of certificates, one for reading a fingerprint and one for reading the iris scan. Again, similarly to BAC, one has to be very careful with multiple EACs – a second failed EAC after a first successful one should bring down the read rights.

6 Passport Architecture and Protocols

By specification, the passport can store up to 19 datagroups, with different information about the holder. Only the first two datagroups are mandatory:

DG1	MRZ info (mandatory)	DG11	Additional Personal Details
DG2	Face image (mandatory)	DG12	Additional Document Details
DG3	Encoded fingerprint	DG13	Optional Details
DG4	Encoded iris scan	DG14	Chip Authentication Keys
DG5	Displayed Portrait*	DG15	Active Authentication Keys
DG6	Displayed Single Digit Fingerprint	DG16	Persons to notify
DG7	Handwritten signature or Usual Mark	DG17	Automatic Border Clearance
DG8	Data Features	DG18	Electronic Visas
DG9	Structure Features	DG19	Travel Records
DG10	Substance Features		

* If present DG5 contains merely a JPEG image of the holder's face. So does DG2, but it also contains additional *face metrics*, like eye or hair color.

Additionally the passport contains up to three special elementary files:

EF.DIR	the "table of contents" file
EF.SOD	the Security Object Directory
EF.CVCA	Terminal Authentication root certificate information

The EF.DIR file merely tells what other files are stored in the passport.

The EF.SOD files is somewhat special and crucial for the document integrity checks. It stores the hashes of all the datagroups, and a signature over all these hashes along with the corresponding country certificate. Checking these stored hashes against the actual datagroup hashes and verifying the signature establishes that the data on the passport had not been tampered with. Verifying the signing certificate against country's issuing authority certificate proves that the passport was indeed issued by the given country. This process is called Passive Authentication (PA). Note that PA cannot prove passport authenticity – cloned passports still pass PA. To prove authenticity one has to perform AA or CA with the passport to establish presence of a genuine private key in the passport.

Finally, the EF.CVCA file stores some information necessary for TA, see below.

In the following we give an in-depth description of the passport authentication protocols, abstracting away the concrete crypto that is used (3DES, RSA, ECC) or the concrete message format. Moreover we used (also in the rest of the paper) reader friendly names for passport APDUs. The actual mapping between our names and proper ISO7816 APDU names [7] is given in Table 2.

select	SELECT FILE
read	READ BINARY
getChallenge	GET CHALLENGE
mutualAuthenticate	EXTERNAL AUTHENTICATE
keyAg	MSE:Set KAT
selectKey	MSE:Set DST, MSE:Set AT
verifyCert	PSO:Verify Certificate

Table 2. ISO7816 APDU names

6.1 BAC

The Basic Access Control protocol is used to establish secure messaging between the passport and the terminal, that is to establish in a secure way session keys sk_{enc} and sk_{mac} for encryption and MAC-ing respectively, as well as the initial value of the sequence counter *ssc*. The sequence diagram for this protocol is given in Figure 4.

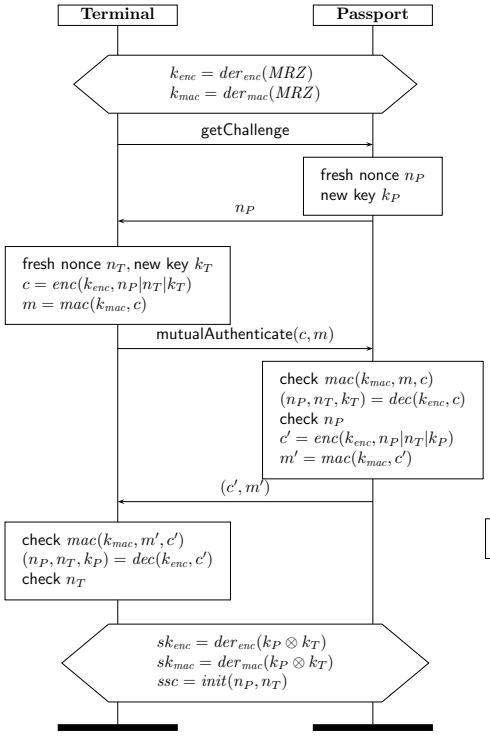


Fig. 4. BAC Protocol

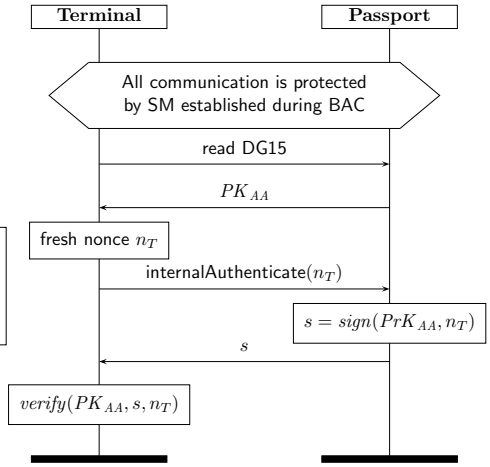


Fig. 5. AA Protocol

To derive the initial static SM keys the terminal has to know the MRZ data of the passport printed on its photo page. This guarantees that the passport is *visually* accessible to the inspecting party (the terminal) and the chip is not being accessed without the owner's consent. During the protocol, the passport and the terminal exchange their nonces (n_P, n_T) and random key seeds (k_P, k_T) using encryption and integrity protection provided by the initial static keys k_{enc} and k_{mac}. After the exchange new session keys are derived and the sequence counter is initialized. From this point on the secure messaging is performed with the new encryption and MAC keys sk_{enc} and sk_{mac}.

6.2 AA

The Active Authentication protocol (Figure 5) can be used to check passports authenticity with a simple challenge-response protocol. The terminal chooses a message n_T to be signed with the passport's public AA key PK_{AA}, which is first retrieved from the passport's datagroup 15. The passport then signs the message with its securely stored private key PrK_{AA} and sends it back to the terminal for verification.

It is now known that this protocol is prone to traceability attacks [4]: there are no limits on the message to be signed by the passport, so e.g. presence of the passport at a certain time and place can be proved by making the passport sign an appropriate message. The Chip Authentication protocol subsumes AA and it rules out such traceability.

6.3 EAC

The Extended Access Control protocol [1, Section 3.1] is somewhat more complicated and, as already mentioned, consists of two steps: Chip Authentication (CA) and Terminal Authentication (TA). During CA, the passport proves its authenticity to the terminal with a Diffie-Hellman key exchange protocol. During TA the terminal then proves to the passport that it has the right to access sensitive biometric data – i.e. the fingerprint or iris scan – by presenting a valid certificate chain. The passport has a root certificate to check validity of the certificate chain.

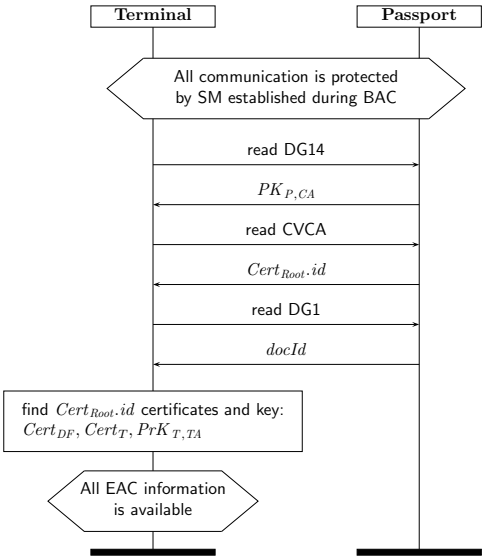


Fig. 6. Preparatory EAC steps

Certificates are issued on a per-country basis. A terminal will typically hold several certificate chains, and it is the nationality of the passport that determines which of these the terminal should provide. So as a preparatory step for EAC, terminal reads the root certificate identifier from the passport's CVCA (Card Verifiable Certification Authority) file, to see if it has a corresponding certificate. The terminal also has to read the passport's document number from datagroup 1 (alternatively from the printed MRZ data) and the passport's public CA key $PK_{P,CA}$ from datagroup 14. These steps are presented in Figure 6.

CA During Chip Authentication [1, Section 3.2] (Figure 7) a single step is used to establish both the chip authenticity and new secure messaging session keys. The underlying cryptographic primitive is Diffie-Hellman key exchange protocol that enables both parties to establish a shared secret (and hence prove to each other the possession of an appropriate private key $PrK_{\_,CA}$) to derive new session keys. After CA is complete, the sequence counter is reset and also both parties record the hash PK_h of the terminal public key $PK_{T,CA}$ to be used during Terminal Authentication.

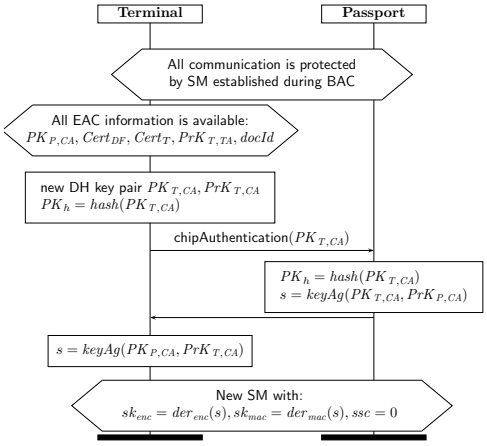


Fig. 7. CA Protocol

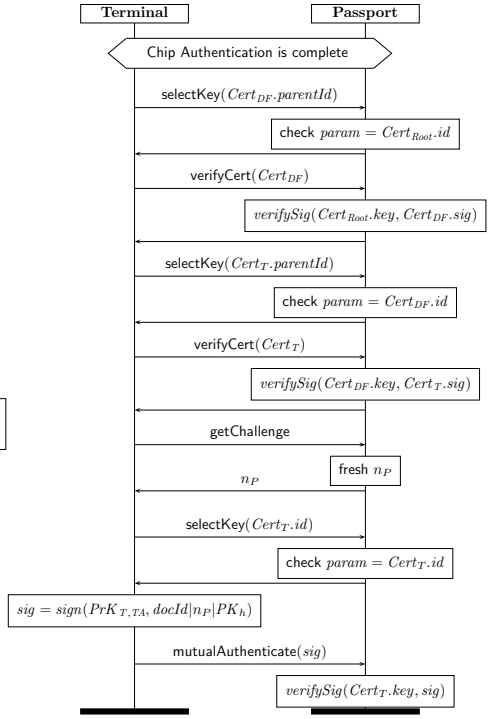


Fig. 8. TA Protocol

TA During Terminal Authentication [1, Section 3.3] the passport receives a chain of certificates from the terminal for verification. Normally two certificates are presented by the terminal. First, the terminal provides a domestic or foreign certificate $Cert_{DF}$ signed by the issuing country's Certification Authority with the root certificate $Cert_{Root}$; the root certificate data (its identifier and the public key)[7] is stored in the passport. Second, the terminal provides a terminal certificate $Cert_T$, signed with the domestic/foreign certificate, for which it holds a corresponding terminal private key $PrK_{T,TA}$.

Before any certificates or signature is presented for verification the terminal selects the corresponding certificate public key on the passport with selectKey. This step is actually redundant, because the passport implicitly knows the key identifier beforehand, it merely serves as an additional consistency check (in fact in the

[7] The TA protocol also allows the terminal to update the current root certificate and current date stored in the passport using special terminal certificates. Accounting for this would complicate our description even more, we refer the reader to [1] for further details.

eDL specification this step is made optional, see below). Then the corresponding certificate signature is verified.

Once the validity of the certificates is established, the terminal authenticates to the passport with a challenge response protocol using its private key $PrK_{T,TA}$ corresponding to the terminal certificate $Cert_T$. First a fresh challenge n_P is received from the passport and then the terminal signs this challenge along with the passport document number and the hash of its public key used during Chip Authentication.

A successful completion of all the above steps grants the terminal the data read rights defined in its terminal certificate $Cert_T$. Note that this does not mean that reading out e.g. a fingerprint from the passport will necessarily be possible, only if the $Cert_T$ access rights say so. The presence of these access rights in this certificate is necessary to issue a certificate that would have a selective right to read out e.g. a fingerprint, but not the iris scan, or vice-versa. A complete TA run, for the case the certificate chain consists of just two certificates, is shown in Figure 8.

7 e-Driving Licenses

The electronic driving license defined in the ISO18013 standard is very similar to the e-passport and our work also applies to driving licenses. The differences from the e-passport are the following. Some of them slightly change the picture presented so far.

7.1 Naming Conventions, Identifiers

The BAC and EAC passport protocols are respectively named Basic Access Protection (BAP) and Extended Access Protection (EAP) in ISO18013. The main distinction between BAP and BAC is that:

- BAP offers more cryptographic configurations. While BAC is offered only in one crypto configuration supporting 3DES encryption and SHA1 hashing, BAP has 4 possible configurations, the strongest one supporting AES-256 encryption and SHA-256 hashing. This difference has no impact on the protocol transitions or access rights.
- The ASN.1[8] object identifiers used in ISO18013 are different from the ones used in ICAO specs. This has no impact on any part of the protocol definitions either.

7.2 More General EAP Access Options

In e-passports the EAC protocol controls access to datagroups 3 (fingerprint) and 4 (iris scan). The values 3 and 4 are "hard coded" into the specification. That is, in the passport no other datagroups can be EAC protected. In the driving license specification this mechanism is made more general. Any valid datagroup (1–24) can be protected by EAP. Here, the consequence is that access control tables, as given in Figure 3 for the passport, have to be more general.

7.3 Optional EAP APDUs

The TA key selection APDU selectKey that is sent to the passport during EAC provide information that is already implicitly known by the passport, namely the identifiers of the certificate with which validity of the next certificate has to be checked. Because of this these steps are optional in the EAP protocol. That is, they can be skipped altogether, or can be sent to the card nevertheless, in which case the

[8] `http://www.asn1.org`

card checks that provided data matches with what it should be. The consequence for our state diagrams is that the steps labelled selectKey in Figure 2 are optional and can be skipped.

8 Conclusions and Future Work

This paper gives an overview of the electronic passport protocols and proposed a way to describe the protocols using state chart diagrams and access control matrices. We believe this provides a useful supplement to the official specs, and might in fact be provided as part of these specs. We ourselves have used these state diagrams to test e-passports using model-based testing [9].

The paper also identifies a number of issues and open questions in the official specifications. Most notably, how an attempt to do a multiple BAC or EAC should be treated is not clear, or how certain failed operations should be handled.

For us the main (practice oriented) methodology in resolving issues like this is to test the product in question, i.e. the passport, to see what the actual implementation does, and consequently what is the common understanding of the specification among the implementors. However, we stress that this should not be used to state what the specification should be, but rather to make statistics on how underspecifications are resolved by implementors. Based on the specification and the security objectives one should determine which freedom is safe for passports to implement, *not* the other way around.

So far we only had a chance to test one EAC enabled passport. Since we were given access to the TA certificates we could test the whole EAC procedure thoroughly, e.g. we found out that on this particular passport one can interleave TA operations with other ones. Without the certificates one can only test the passport until a successful CA. Obviously, we would like to get more opportunities to fully test different EAC implementations in future. As for the e-driving license, because the standard is very recent (it has been finalised at the beginning of 2009) it seems that our open source implementation is the only one available for inspection.

Another ongoing project is automatic development of our state diagrams using automatic learning techniques [2]. Here a passport is probed with different commands and by interpreting the passport responses a state chart diagram for the protocol is build. This will allow fast and automatic discovery of particular choices in a given passport implementation.

References

1. Advanced security mechanisms for machine readable travel documents – Extended Access Control (EAC) – Version 1.11. Technical Report TR-03110, German Federal Office for Information Security (BSI), Bonn, Germany, 2008.
2. Fides Aarts, Bengt Jonsson, and Johan Uijen. Generating models of infinite-state communication protocols using regular inference with abstraction. Submitted.
3. Tom Chothia and Vitaliy Smirnov. A traceability attack against e-passports. In *14th International Conference on Financial Cryptography and Data Security 2010*, LNCS. Springer, 2010. To appear.
4. Jaap-Henk Hoepman, Engelbert Hubbers, Bart Jacobs, Martijn Oostdijk, and Ronny Wichers Schreur. Crossing borders: Security and privacy issues of the European e-Passport. In *Proc. IWSEC 2006: Advances in Information and Computer Security*, number 4266 in LNCS, pages 152–167. Springer, 2006.
5. Doc 9303 – Machine readable travel documents – Part 1–2. Technical report, ICAO, 2006. Sixth edition.
6. Supplement to Doc 9303. Technical report, ICAO, November 2008. Release 7 – Final.

7. ISO 7816. ISO/IEC 7816 Identification cards – Integrated circuit(s) cards, Part 4: Organization, security and commands for interchange. Technical report, ISO JTC 1/SC 17, 2005.
8. Jean Monnerat, Serge Vaudenay, and Martin Vuagnoux. About machine-readable travel documents: Privacy enhancement using (weakly) non-transferable data authentication. In *International Conference on RFID Security 2007*, pages 15–28, 2007.
9. Wojciech Mostowski, Erik Poll, Julien Schmaltz, Jan Tretmans, and Ronny Wichers Schreur. Model-based testing of electronic passports. In M. Alpuente, B. Cook, and C. Joubert, editors, *Formal Methods for Industrial Critical Systems 2009, Proceedings*, volume 5825 of *LNCS*, pages 207–209. Springer, November 2009.
10. Martijn Oostdijk, Dirk-Jan van Dijk, and Maarten Wegdam. Usercentric identity using epassports. In *Security and Privacy in Communication Networks*, volume 19 of *LNICST*, pages 296–310. Springer, 2009.
11. Hening Richter, Wojciech Mostowski, and Erik Poll. Fingerprinting passports. In *NLUUG Spring Conference on Security*, pages 21–30, 2008.