Automated Passport Control: Mining and Checking Models of Machine Readable Travel Documents

Stefan Marksteiner
stefan.marksteiner@avl.com
AVL List Gmbh
Graz, Austria
Mälardalen University
Västerås, Sweden

Marjan Sirjani
marjan.sirjani@mdu.se
Mälardalen University
Västerås, Sweden

Mikael Sjödin
mikael.sjodin@mdu.se
Mälardalen University
Västerås, Sweden

ABSTRACT

Passports are part of critical infrastructure for a very long time. They also have been pieces of automatically processable information devices, more recently through the ISO/IEC 14443 (Near-Field Communication – NFC) protocol. For obvious reasons, it is crucial that the information stored on devices are sufficiently protected. The International Civil Aviation Organization (ICAO) specifies exactly what information should be stored on electronic passports (also Machine Readable Travel Documents – MRTDs) and how and under which conditions they can be accessed. We propose a model-based approach for checking the conformance with this specification in an automated and very comprehensive manner: we use automata learning to learn a full model of passport documents and use trace equivalence and primitive model checking techniques to check the conformance with an automaton modeled after the ICAO standard. Since the full behavior is underspecified in the standard, we compare a part of the learned model and apply a primitive checking ruleset to assure proper authentication. The result is an automated (non-interactive), yet very thorough test for compliance, despite the underspecification. This approach can also be used with other applications for which a specification automaton can be modeled and is therefore broadly applicable.

CCS CONCEPTS

• **Security and privacy → Systems security**; **Penetration testing**; • **Software and its engineering → Software verification and validation**.

KEYWORDS

NFC, Automata Learning, Protocol Compliance, Bisimulation, Formal Methods, Passports

ACM Reference Format:
Stefan Marksteiner, Marjan Sirjani, and Mikael Sjödin. 2024. Automated Passport Control: Mining and Checking Models of Machine Readable Travel Documents. In *The 19th International Conference on Availability, Reliability and Security (ARES 2024), July 30-August 2, 2024, Vienna, Austria.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3664476.3670454

ARES 2024, July 30-August 2, 2024, Vienna, Austria

ACM ISBN 979-8-4007-1718-5/24/07…$15.00
https://doi.org/10.1145/3664476.3670454

1 INTRODUCTION

Passports are among critical infrastructure and subject to forgery for a very long time. This has been aggravated by the fact – it is even mandatory for member states of the International Civil Aviation Organization (ICAO) that all documents that are not machine-readable travel documents (MRTDs) are expired since 2015[1]. More recently, passports have opened up for wireless reading via Near-Field Communication (NFC). The NFC communication protocol specified in the ISO/IEC 14443 standard series (most prominently in ISO/IEC 14443-4 [9] for data communications), while the respective commands for interacting with integrated-circuit identification cards are defined in ISO/IEC 7816-4 [10]. The ICAO Doc 9303 series specifies a logical data structure and further details regarding commands, as well as rule sets for accessing the data inside the defined structure for MRTDs [16]. To sum it up, we use automata learning with SELECT, READ, UPDATE and AUTHENTICATE (implementing Basic Access Control - BAC [7]) symbols from the ISO/IEC 14443-4 protocol to infer an automaton and compare it using trace and bisimilarity equivalence to an automaton modeled after the ICAO MRTD specification. The remainder of this paper is structured as follows: Section 1.1 contains this paper's additions to the body of knowledge, Section 2 outlines the necessary prerequisite knowledge, Section 3 describes the learning and conformance checking setup, Section 4 gives evaluation results, Section 5 gives an overview of important related work, and Section 6 concludes the paper with a discussion and an outlook on further research directions.

1.1 Contribution

This paper outlines a process how to very thoroughly analyze passports and automatically check their conformance with the ICAO MRTD specifications using formal methods. The contribution to the body of knowledge is threefold. The paper provides:

- A concise summary of the ICAO MRTD specification – this information can be used by researchers to build compliance checking systems.
- A(n incomplete) state machine model of the specification.
- An automata learning setup for ISO/IEC 14443-4, ISO/IEC 7816-4, and ICAO Doc 9303 including an input alphabet definition and a practical implementation.

[1] https://www.icao.int/Newsroom/Pages/Last-Week-for-States-to-Ensure-Expiration-of-Non-Machine-Readable-Passports.aspx

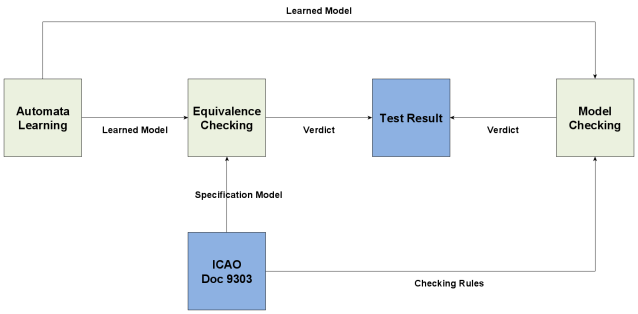

Figure 1: Overview of the approach. Green are processes and blue are artifacts.

- A practically implemented method for compliance checking based on equivalence and primitive model checking for an underspecified standard.

The approach described in this paper can also be adapted, e.g., by using the specification parts model (Sections 2.2, 3.3, and 3.4) to create model-based tests from it (removing the learning part) or by using the learning part only (Sections 3.1, 3.2, and 4) and creating rules for a model checker to check the model for desired properties (removing the equivalence checking part). Figure 1 provides an overview of this approach and its parts.

1.2 Limitations

Since the ICAO specification is not strict enough to build a feasible specification automaton (through many optionalities, there could be many automata that represent a valid model of the specification), the behavioral equivalence only covers a part of the conformance checking. For other parts (namely, files should not be read and/or writable if certain conditions are not met), we impose model checks. These are, however, pretty primitive and might be improved. Also, with the lack of a Password Authenticated Connection Establishment (PACE) and Terminal Authentication implementation, we do not check for these types of authentication [2]. Lastly, the case study is limited, as only Austrian passports were available as examination objects.

2 PRELIMINARIES

This section outlines the fundamentals of the NFC protocol family, the ICAO MRTD specifications and the concepts of automata learning.

2.1 Near Field Communication

Near Field Communication (NFC), standardized in the ISO/IEC 14443 series, is a wireless communications protocol that allows for reader devices (proximity coupling devices - PCDs) to communicate completely passive, powerless devices (proximity integrated circuit cards - PICCs) at low data rates (up to 424 kbits/s). The PICCs are therby powered via an inductive field that also transports the data. After a handshake [8], which is out of scope of this paper, communications is standardized in ISO/IEC 14443-4 [9], which defines basic types of messages for data transmission (information or *I* blocks), signaling (supervisory or *S* blocks), and acknowledgements (receive-ready or *R* blocks), along with protocol mechanisms like block numbering, chaining, error correction, etc. The actual data interchange format is defined in ISO/IEC 7814-4 [10], including commands for data selection, data manipulation and security functions. In general, data on NFC cards is segmented into different applications (*dedicated files - DFs*), which are comparable to directories on file systems, that contain data files (*elementary files - EFs*) as actual data storages. Both types are selected using the *SELECT* command with different parameters. The standard also defines manipulation operators that can be applied on EFs. One flavor are the *READ/WRITE/UPDATE/APPEND/SEARCH/ERASE/COMPARE BINARY* commands. These are for manipulating *Data Units*, which can inside the EF be controlled by using offsets. Another set is the *READ/WRITE/UPDATE/APPEND/SEARCH/ERASE/ACTIVATE/DEACTIVATE RECORD* commands for manipulating *Records*, that can be addressed by record identifiers instead of raw binary offsets. Under records reside *Data objects (DOs)*, which can be addressed with the *GET/PUT/UPDATE/COMPARE DATA* commands. Apart from that, the standard defines security functions like the *GETCHALLENGE* and different forms of *AUTHENTICATE* and *verification* commands. These commands build the base of the input alphabet for learning NFC models, the rest of the commands inside an abstraction layer will be defined by respective identifiers defining the content.

The respective answers to these commands could consist of data (which can be encrypted as well), but in any case contains two status bytes (as defined in ISO/IEC 7816-4). These bytes are set to have a scheme distinguishing between a completed process with normal processing (*9000* and *61XX* - the latter means that are data bytes left to transmit) or warning processing (*62XX* and *63XX*), as well as aborted processing with execution error (*64XX* and *66XX*) or checking eroor (*67XX* and *6FXX*). The most common codes encountered when working with passports are (empirical :

- *9000* - OK
- *6300* - No information given (seen at authentication attempts with wrong credentials)
- *6700* - Error with no information given (when trying to perform write operations without authentication)
- *6982* - Security status not satisified (i.e., lack of authentication)
- *6985* - Conditions of use not satisfied (when trying to authenticate without an application selected)
- *6986* - Command not allowed (when trying to read without a file selected)
- *6988* - Insecure messaging DOs (when encrypting data with a wrong key)
- *6A82* - File not found
- *6D00* - Instruction code not supported or invalid (when sending malformed commands)

[2] A German passport was available, but supported PACE only. We could not include it in the evaluation for the reason stated above.

2.2 Machine Readable Travel Document Specification

The International Civil Aviation Organization's (ICAO) Doc 9303 series specifies the appearance and behavior of passports and machine readable travel documents (MRTDs). Particularly Part 10 [16] specifies the logical data structure (LDS) of MRTDs and defines access rights (i.e. what authentication is necessary to read or manipulate) for data.

The standard defines four applications, referenced by dedicated files:

- eMRTD (ID A0 00 00 02 47 10 01)
- Travel records (ID A0 00 00 02 47 20 01)
- Visa records (A0 00 00 02 47 20 02)
- Additional biometrics (A0 00 00 02 47 20 03)

The first (following the LDS1) is mandatory, while the latter three (follwing the LDS2) are optional. The eMRTD application contains all of the data that is normally on the main page of a passport (like number, name, birth date, expiration date, etc.) plus additional data including electronic photos, finger and iris scans. This application contains data that should be immutable in the document and readable with authentication, namely with the older Basic Access Control (BAC) or the newer Password Authenticated Connection Establishment (PACE), with sensitive biometrics (fingerprints and iris scan) additionally needs a terminal authentication to determine the reader is authorized. Due to this is mandatory and the other applications are not implemented in many (including EU) passports[3], we concentrate on this part. The other applications contain potentially mutable records, and certificates stored within the application to display authenticity against a reader – assuring that visa and electronic travel stamps are genuine. The applications require different levels of authentication (see Section 2.2). Figure 2 gives an overview of the layout and the different applications and Table 1 gives an overview of the defined EFs with their IDs, DFs, requisiteness, and access requirements.

2.2.1 Electronic Machine Readable Travel Document Application. Concentrating on the Electronic Machine Readable Travel Document (eMRTD) application, ICAO Doc 9303 Part 10 defines various EFs that contain personal and document data, along with access requirements. In particular it defines

- *Common (EF.COM)*: containing metadata (version, encoding, etc.) of the application
- *Data Group 1 (EF.DG 1)*: containing the machine readable zone.
- *Data Group 2 (EF.DG 2)*: containing the holder's face image.
- *Data Group 3 (EF.DG 3)*: containing the holder's fingerprints image.
- *Data Group 4 (EF.DG 4)*: containing the holder's iris image.
- *Data Group 5 (EF.DG 5)*: containing holders displayed portrait(s).
- *Data Group 6 (EF.DG 6)*: is reserved for future use.
- *Data Group 7 (EF.DG 7)*: containing the holder's displayed signature.
- *Data Group 8 (EF.DG 8)*: containing data features.
- *Data Group 9 (EF.DG 9)*: containing structure features.
- *Data Group 10 (EF.DG10)*: containing substance features.
- *Data Group 11 (EF.DG11)*: containing additional personal details (e.g., localized name, place-of-birth).
- *Data Group 12 (EF.DG12)*: containing additional document details (e.g., issuing authority, date-of-issue).
- *Data Group 13 (EF.DG13)*: containing optional details.
- *Data Group 14 (EF.DG14)*: containing data elements.
- *Data Group 15 (EF.DG15)*: containing the public key info for active authentication.
- *Data Group 16 (EF.DG16)*: containing persons to notify.
- *Document Security Object (EF.SOD)*: containing hash values of the data group for integrity checking.
- *Country Verifying Certification Authorities (EF.CVCA)*: containing public keys of CVCA for teminal authentication (see Section 2.2.2).
- Key files for authentication (see Section 2.2.2).

All of these files can be mandatory (DG1, DG2), optional (DGs 3-5, 7-13, and 16), or conditional (DG14 - if PACE is implemented, DG15 - if AA is implemented) and can be read if authenticated (via BAC or PACE), except for DGs 3 and 4, which require additional terminal authentication (see Section 2.2.2) - none of these files should be manipulated (no write/append access). Table 1 gives an overview of these files, along with those from the LDS2 applications.

2.2.2 Authentication. Since passports can considered critical infrastructure devices, authentication is crucial. The ICAO defines two mechanisms for access to the MRTD chips in its Doc 9303-11 standard [7]:

- Basic Access Control (BAC) and
- Password Authenticated Connection Establishment (PACE).

BAC is the older one, it had known privacy issues [2, 5] and may become deprecated in future. Currently an MRTD must implement one or both mechanisms. Additionally LDS2 applications *must* and additional biometrics, LDS1 data groups 3 (fingerprints) and 4 (iris), *may* be secured by a terminal authentication procedure.

Since we have a BAC, but not a PACE implementation available and our available devices-under-test (see Section 4) all support BAC but only partially PACE, we use BAC only for modeling and evaluation. BAC uses is challenge-response by encrypting a received nonce (via a *GETCHALLENGE* command) with a key derived from three components[7]: the passport number, the expiration date and the holder's birth date. These can be obtained from the machine readable zone or the main page of the passport. The encrypted nonce is subsequently sent through an *EXTERNAL AUTHENTICATE* command to finish the authentication. The authentication answer contains additional key material for establishing session keys. All instructions (and respective responses) operating on data protected by this particular BAC are encrypted using these session keys. The rationale is to prevent unnoticed wireless data extraction from an MRTD.

2.3 Automata Learning

Automata learning is a method to infer state machine models (originally deterministic finite acceptors - DFAs) from a system using a

[3]We also concretely tested an expired Austrian and a valid Austrian and German passport for these applications. The answer was unanimously the status code *6A82* for *File or application not found.*

Name	ID	DF	Mandatory	SELECT	READ	WRITE	APPEND
EF.ATR/INFO	2F01	Master	No[1]	ALWAYS	ALWAYS	NEVER	NEVER
EF.DIR	2F00	Master	No[1]	ALWAYS	ALWAYS	NEVER	NEVER
EF.CardAccess	011C	Master	No[2]	ALWAYS	ALWAYS	NEVER	NEVER
EF.CardSecurity	011D	Master	No[2]	PACE	PACE	NEVER	NEVER
EF.DG1,2	0101,02	LDS1.eMRTD	Yes	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.DG3,4	0103,04	LDS1.eMRTD	No	BAC/PACE+TA	BAC/PACE+TA	NEVER	NEVER
EF.DG5,7-13,16	0105,07-D,10	LDS1.eMRTD	No	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.DG6 (RfFU)	0106	LDS1.eMRTD	No	-	-	-	-
EF.DG14	010E	LDS1.eMRTD	No[2]	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.DG15	010F	LDS1.eMRTD	No[3]	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.COMMON	011E	LDS1.eMRTD	Yes	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.SOD	011D	LDS1.eMRTD	Yes	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.CVCA	011C	LDS1.eMRTD	Yes	BAC/PACE	BAC/PACE	NEVER	NEVER
EF.Certificates	011A	All LDS2	No	PACE+TA	PACE+TA	NEVER	PACE+TA
EF.ExitRecords	0102	LDS2.Travel Records	No	PACE+TA	PACE+TA	NEVER	PACE+TA
EF.EntryRecords	0101	LDS2.Travel Records	No	PACE+TA	PACE+TA	NEVER	PACE+TA
EF.VisaRecords	0103	LDS2.Visa Records	No	PACE+TA	PACE+TA	NEVER	PACE+TA
EF.Biometrics1-64	0201-0240	LDS2.Add. Biometrics	No	PACE+TA	PACE+TA	NEVER	NEVER

Table 1: Files from ICAO Doc 9303-10 with their names, IDs, application, requisiteness and access requirements.
[1]Conditional - required if LDS2 files are present.
[2]Conditional - required if PACE is implemented.
[3]Conditional - required if active authentication is implemented.

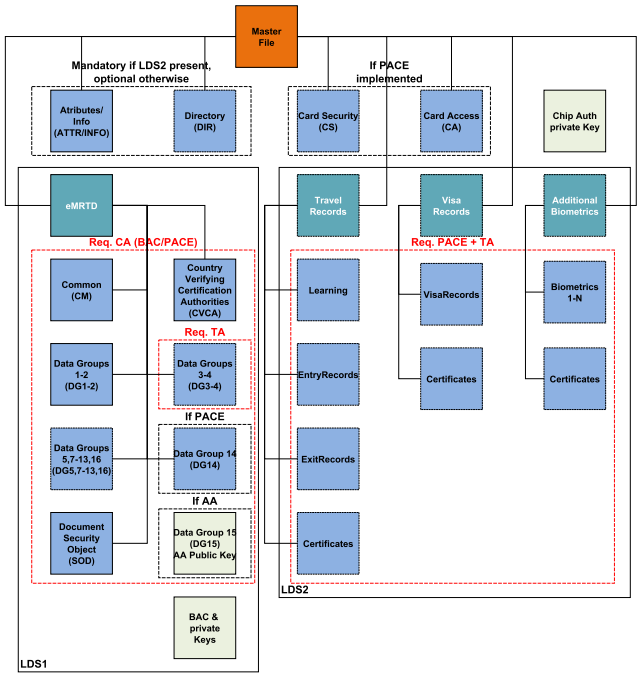

Figure 2: Locical Data Structure of Machine Readable Travel Documents.
Amber is the master file (MF), Cyan are dedicated files (DF), Blue are Elementary Files (EF), and Green are key files. Solid frames means mandatory files, dashed ones optional files. Solid boxes donate the LDS contexts, dashed black boxes requirements, and dashed red boxes necessary authentication.

learner-teacher framework [1]. The learner may ask two kinds of questions: *Membership queries* and *Equivalence queries*. The former is used to determine if an input (specifically an input word, which is a combination of input symbols) is *well-formed* i.e., if it is a valid word inside this language. The answer to this query is a yes or a no from the teacher. The latter type of queries is to determine the correctness of a learned automaton. The teacher answers with yes if a hypothesis (inferred after a sufficient amount of membership queries) is correct or gives a counter example the learner could use to improve the hypothesis until it is correct. To apply this to real-world reactive systems (mostly software or cyber-physical systems), we use Mealy machines (formally $M = (Q, \Sigma, \Omega, \delta, \lambda, q_0)$, with Q being a set of states, Σ an input alphabet, Ω an output alphabet, δ a transition function ($\delta : Q \times \Sigma \rightarrow Q$), λ an output function ($\lambda : Q \times \Sigma \rightarrow \Omega$) and q_0 an initial state. This alters the framework in the sense that the membership queries yield a Mealy output instead of a binary answer. In practice, the teacher is implemented in a way that the membership queries deliver the (abstracted - see Section 3.1) result directly obtained from the reactive system and the equivalence queries are realized as set of conformance tests that deliver a positive answer or a counterexample in the form of a failed test's input. The respective systems can also be viewed as labeled transition systems (LTS) [14]. We can therefore use trace equivalence ($Traces(LTS_1) = Traces(LTS_2)$) to check the behavioral equivalence with a specification automaton [3].

3 LEARNING SETUP

We use the widely used Java library LearnLib [12] to mine (Mealy type) state machine models of passports. This library provides classes for developing adapters to a system-under-learning (SUL),

as well as various learning algorithms (L* [1] and variants thereof [17], KV [13], DHC [15] and TTT [11]).

The SUL classes interact with a C++ program that serves as an interface for a Proxmark3 NFC adapter device [6], which allows for sending arbitrary NFC commands to a device and process the respective responses. This program also contains an abstraction layer (see Section 3.1).

3.1 Abstraction

Since, in principle, any combination of bytes can be sent to a SUL, the input space is very large (only bounded by the maximum transmission units for NFC). To keep the learning within a feasible time frame, the input must be limited to sensible set of discrete instructions i.e., the input alphabet for a Mealy machine. The C++ adapter translates input symbols to data to be send. Similarly, we abstract the output of the operations. This is a necessity, since some of the commands yield a different output every time (e.g., through random cryptographic nonces, session keys, etc.). However, conviently all of the answer messages contain a status code (see Section 2.1), which is even in clear text for encrypted messages. Also, the status code already contains the relevant information for checking ICAO conformance, since it determines whether a file could be successfully read or manipulated. We therefore use the answer status codes as abstracted outputs.

3.2 Input Alphabet

The input alphabet in our case consists of a combination of instructions from ISO/IEC 7816-4 (select DF, select EF, GETCHALLENGE, EXTERNAL AUTHENTICATION, READ BINARY, and UPDATE BINARY) and the file structure with the DF and EFs outlined in Section 2.2. Except for the BAC, all instructions are used in two forms: unencrypted and encrypted. The rationale is to check if after a successful authentication insecure access might become possible. We use READ BINARY as representative for all reading operations and UPDATE BINARY as representative for writing operations. As stated above we concentrate on the LDS1 application, making this the only select DF instruction. The codes for card access (CA) and (CVCA), as well as for card security (CS) and the document security object (SOD) are identical (only executed in different context) yielding to only one encrypted and unecrypted input symbol for each. The BAC is abstracted in to one input symbol (combining the GETCHALLENGE and EXTERNAL AUTHENTICATE instructions along with all necessary key calculations). The compelete input alphabet is therfore <SEL_EF.CA>, <SEL_DF.LDS1>, <SEL_EF.CM>, <SEL_EF.DG1>, <SEL_EF.DG2>, <SEL_EF.DG3>, <SEL_EF.DG4>, <SEL_EF.DG5>, <SEL_EF.DG6>, <SEL_EF.DG7>, <SEL_EF.DG8>, <SEL_EF.DG9>, <SEL_EF.DG10>, <SEL_EF.DG11>, <SEL_EF.DG12>, <SEL_EF.DG13>, <SEL_EF.DG14>, <SEL_EF.DG15>, <SEL_EF.DG16>, <SEL_EF.SOD>, <SEL_EF.ATR>, <SEL_EF.DIR>, <RD_BIN>, <BAC>, <SSEL_EF.CA>, <SSEL_DF.LDS1>, <SSEL_EF.CM>, <SSEL_EF.DG1>, <SSEL_EF.DG2>, <SSEL_EF.DG3>, <SSEL_EF.DG4>, <SSEL_EF.DG5>, <SSEL_EF.DG6>, <SSEL_EF.DG7>, <SSEL_EF.DG8>, <SSEL_EF.DG9>, <SSEL_EF.DG10>, <SSEL_EF.DG11>, <SSEL_EF.DG12>, <SSEL_EF.DG13>, <SSEL_EF.DG14>, <SSEL_EF.DG15>, <SSEL_EF.DG16>, <SSEL_EF.SOD>, <SSEL_EF.ATR>, <SSEL_EF.DIR>, <SRD_BIN>.

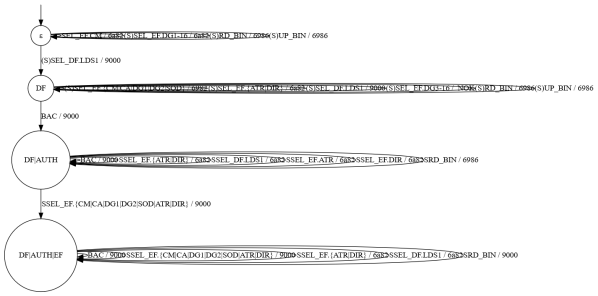

Figure 3: Graph of the specification automaton. The diagram was simplified for better readability.

Using this alphabet in the abstracted learner yields models of passports (see Section 4). This model can be used to compare with a specification automaton or to use with model checking.

3.3 Specification Automaton

Following the specification, we modeled a minimal automaton that is able to behave a required by ICAO specification Doc 9303-10. This is not straight-forward, since, compared with the possible NFC command alphabets, the ICAO document is under-specified. Additional behavior is specified in Doc 9303-11 (particularly that the LDS1.eMRTD application *must* be selected before performing BAC authentication). Since much of the behavior is not defined by ICAO (e.g., behavior on multiple authentications and selections), we modeled the bare functional minimum. The automaton contains 5 states, which, for better comprehensiveness, we label according to already happened events: the initial state (ε), a state with a valid EF from the Master DF (*EF*), the LDS1 application selected (*DF*), the LDS1 application selected and authenticated with BAC (*DFAUTH*), and the LDS1 application selected, authenticated with BAC and a valid EF from the LDS1 application selected *DFAUTHEF*. Since some files are optional we also leave out the selection transitions in the state where they could be successfully accessed, since the respective transitions could have a positive (*9000*) or negative (*6a82*) output. Figure 3 show a graph of the specification automaton. Note that the automaton is not a complete one, as not every input has a defined output and target state in the output and transition relations, respectively. This yields a specification automaton that can be used for either a learning and behavior comparison approach as described in this paper, or to create model-based tests.

3.4 Simplification and Labeling

To allow for a sensible comparison of the learned model with the specification, we want to identify the distinct states with a hard definition in the standard in order to separate it from not defined parts, as not the full behavior but only the access rules and requiredness of files are prescribed (see Section 2.2). Also, semantically labeled states are more convenient for both the human reader and electronic processing. We therefore use a simple algorithm that accesses mandatory files in a specific order, namely:

- From the initial state, follow the select EF for CardAccess transition
- If output yields 9000, label this state as *EF*

- From the inital state, select the DF for LDS1.eMRTD tansition
- If output yields 9000, label this state as *DF*
- From DF, follow the BAC transition
- If output yields 9000, lable this state as *DF|AUTH*
- From DF|AUTH, follow the encrypted select Data Group 1 transition
- If output yields 9000, label this state as *DF|AUTH|EF*
- From the initial state, select the BAC transition
- If output yields 6985 label the state as *FAILAUTH*
- From FAILAUTH, follow the select LDS1.eMRTD transition
- If output yields 9000 label the state as *FAILAUTH|DF*
- From EF, select the BAC transition
- If output yields 6985 label the state as *FAILAUTH|EF*
- From the DF|AUTH|EF, select the unencrypted READ BINARY transition
- If output yields 6982 label the state as *DEAUTH*

This names the states after attributed properties: *EF* for a selected elementary file, *DF* for a selected decicated file (i.e. the LDS1.eMRTD application), AUTH for a successful authentication (i.e., BAC), FAILAUTH for a failed authentication (mainly happens because no file that needs authentication was selected beforehand), *DEAUTH* for a revoked authentication (which occurs when insecure – i.e., non-encrypted – commands are executed). This label states where taken from different Austrian passports.

3.5 Specification Conformance

As stated above, through underspecification there is room for diverse behavior patterns. As only the access level and optionality of files is defined, we created a minimal automaton that modeled the access rules for present files. This automaton only contains the ε, *DF*, *DF|AUTH*, and *DF|AUTH|EF* states. File operations (i.e. *READ BINARY* should only be possible in the *DF|AUTH|EF* state. We therefore abstracted the output into successful operations (*9000*) and unsuccessful operations (*NOK*) for optional files in the other states. It is, however, not significant for an optional EF if access to it has been denied because of lack of authentication or because the file is not present. Since we identified 22 different non-mandatory EFs (all inside the Master and the eMRTD DFs), that would otherwise have led to 2^{22} possibilities equalling just as many specification automata. Inside the *DF|AUTH|EF* we left the transitions out for optional files (since, according to the specification, an operation may or may not successful) and modeled successful read operations for mandatory ones. For conformance checking, two steps were necessary: a) positive checking and b) negative checking.

For a) we used a trace equivalence check with the specification automaton, removing any states and transitions from the learned one that are not in the specification. The pracical implementation is realized by removing all transitions from a learned automaton that lead from or to a state that is not covered within the specification automaton (which, again is not complete and is missing ambiguous transitions, i.e. such for optional files). For the remainder we perform a trace equivalence check between the learned and the specification automata. We realize this by converting the LearnLib output in the Graphviz (.dot) format into the Aldebaran (.aut) format and feed it into the mCRL2 tool [4] for trace equivalence checking. However, all non-covered transitions will be removed in an examined learned automaton as well, so the trace equivalence shall hold if the SUL conforms.

For b) we performed a primitive form of model checking using a simple rule set:
(i) Since for all mandatory files reside in the LDS1.eMRTD application and authentication is required for access, a READ BINARY must be secured (*SRD_BIN*) and executed for from the *DF|AUTH|EF* state) to yield a positive result (*9000*).
(ii) For *DF|AUTH|EF* to be in a *guaranteed* authenticated state, every transition targeting that state must come from *DF|AUTH* or come through a successful authentication (*BAC / 9000*) transition[4]. This enables for efficient, automatic conformance checking that is as comprehensive as the specification allows.

4 EVALUATION

We put the methodology to test at two different passports from the Republic of Austria: one current and one expired 5 years ago. We were able to infer models of these passports and observed subtle differences, particularly that the elder one obviously does not support PACE (missing the respective CardAccess and CardSecurity files in the master record). Figure 4 shows a diagram of the automaton of the current passport, which is simplified for readability but a full model in the sense that it is not the reduced version used for conformance checking as outlined in Section 3.5. The model shows additional states as compared to the (partial) specification, namely a de-authenticated state, as well as failed authentications and combinations of this new and the other states. Concretely, the additional states are *EF* (a selected Elementary File in the Master file), *FAILAUTH*, *FAILAUTH|EF*, *FAILAUTH|DF*, and *DEAUTH* (a de-authentication after a wrongly selected EF in the *DF|AUTH|EF* state. The main task for a checker is to make sure that no illegal operation (i.e., reading or manipulation operation) occurs in these states. Both examined objects passed the conformation tests (equivalence and simple model checking) as outlined in Section 3.5.

5 RELATED WORK

We used the approach of using equivalence checking (bisimulation and trace equivalence) with NFC before, particularly for an automatic compliance checker for the ISO/IEC 14443-3 (the NFC handshake) protocol [14]). Apart from that, usage of similar approaches for compliance checking is sparse. For Mealy Machines, Tappler et al. [19] used bisimulation for comparison and there is work for a similar approach checking an embedded control software for its correctness [18].

6 CONCLUSION

In this paper, we demonstrated the usage of automata learning to infer models of passports and check whether they comply to international machine readable travel document specification. We therefore distilled the relevant information to create a specification automaton out of the relevant documents and modeled a labeled transition system out of it, which contains the standard-compliant behavior. Since many of the files are not mandatory, we abstracted the output

[4]While other possibilities are possible in principle, they cannot be guaranteed to be conform with the standard; as a de-authentication might have occurred, the authentication can not be take for granted.

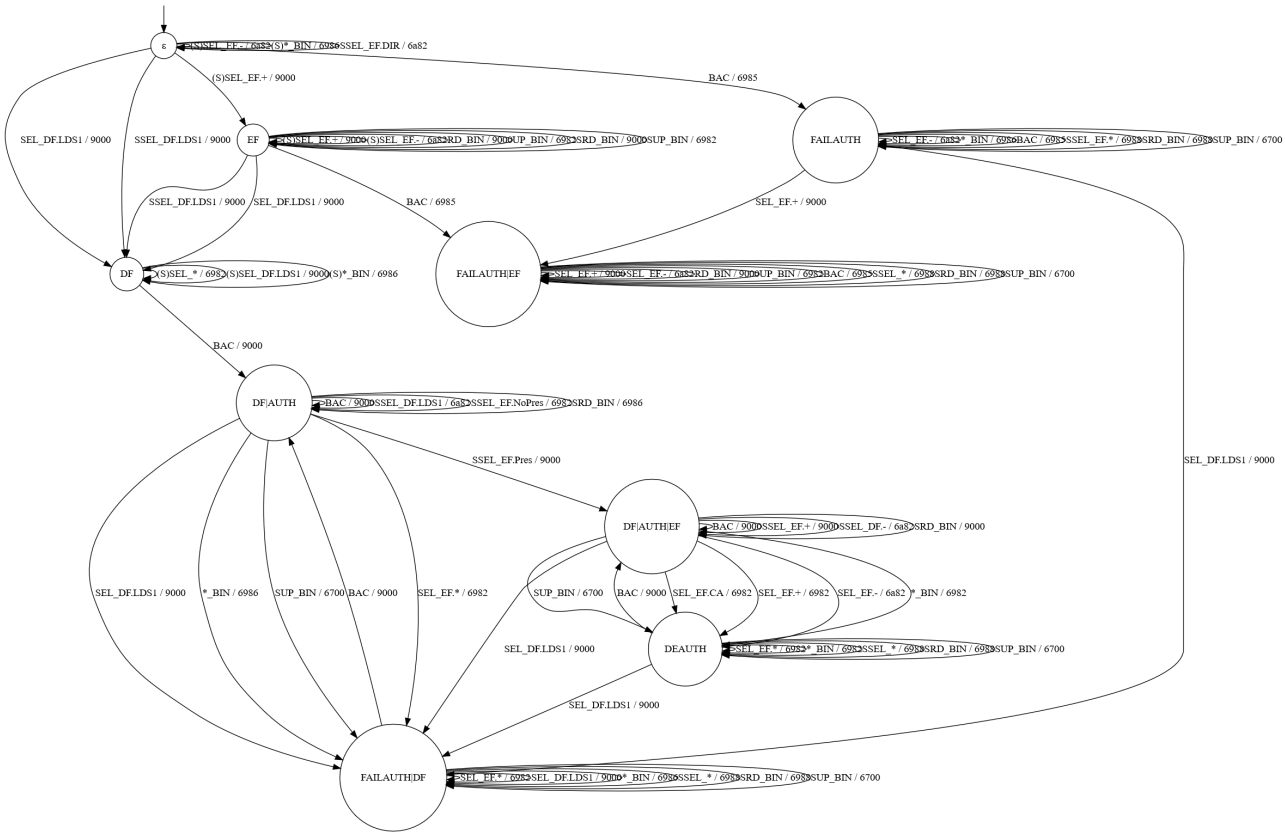

Figure 4: Learned and labeled model of an Austrian passport. The diagram was simplified for better readability.

for the comparison of optional files outside the state where a file was selected and authentication successful. We therefore coped with an underspecified standard using incomplete automata. This system was compared with a learned model of an actual passport, using the input alphabet described in Section 3.2. Since the models displayed equivalent behavior, evidence for conformance with the ICAO standard was provided.

6.1 Discussion

The current implementation is not complete and has therefore some limitations. Most prominently, no other authentication mechanisms than BAC was implemented, therefore parts of the specification (particularly PACE-related) could not be tested – this also ruled out some newer passports, e.g., current German ones, as systems-under-test, as BAC can be completely abandoned as authentication mechanism in favor of PACE. Also, no systems containing LDS2 applications were available, so these could also not be tested.

6.2 Outlook

The methods in this paper provide principally a very thorough method of NFC-based data systems (particularly passports). This method can easily adapted to be used with other systems and protocols, once provided with an adequate specification, learner adapter and input alphabet. Another direction to move forward is to test more specifically: instead of checking equivalent behavior with a specification automaton, specific rules can be applied for a model checker to check the system for certain properties (particularly, security properties). Looking in another direction, the specification models are created manually so far. To further automate the process techniques like Natural Language Processing (NLP) using small or large language models can be used to create specifications automata from standards or specification documents. This also facilitates the use case of Original Equipment Manufacturers (OEMs) being able to very thoroughly examine the

ACKNOWLEDGMENTS

This research received funding within the CHIPS Joint Undertaking (JU) under grant agreement No. 101007350 (project AIDOaRt). The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey. The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. *Information and Computation* 75, 2 (Nov. 1987), 87–106. https://doi.org/10.1016/0890-5401(87)90052-6

[2] Myrto Arapinis, Vincent Cheval, and Stéphanie Delaune. 2015. Composing Security Protocols: From Confidentiality to Privacy. In *Principles of Security and Trust* (Berlin, Heidelberg), Riccardo Focardi and Andrew Myers (Eds.). Springer, 324–343. https://doi.org/10.1007/978-3-662-46666-7_17

[3] Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking.* MIT Press.

[4] Olav Bunte, Jan Friso Groote, Jeroen J. A. Keiren, Maurice Laveaux, Thomas Neele, Erik P. de Vink, Wieger Wesselink, Anton Wijs, and Tim A. C. Willemse. 2019. The mCRL2 Toolset for Analysing Concurrent Systems. In *Tools and Algorithms for the Construction and Analysis of Systems (Lecture Notes in Computer Science)*, Tomáš Vojnar and Lijun Zhang (Eds.). Springer International Publishing, Cham, 21–39. https://doi.org/10.1007/978-3-030-17465-1_2

[5] Tom Chothia and Vitaliy Smirnov. 2010. A Traceability Attack against E-Passports. In *Financial Cryptography and Data Security* (Berlin, Heidelberg), Radu Sion (Ed.). Springer, 20–34. https://doi.org/10.1007/978-3-642-14577-3_5

[6] Flavio D Garcia, GT de Koning Gans, and Roel Verdult. 2012. *Tutorial: Proxmark, the Swiss Army Knife for RFID Security Research: Tutorial at 8th Workshop on RFID Security and Privacy (RFIDSec 2012).* Technical Report. Radboud University Nijmegen, ICIS, Nijmegen.

[7] International Civil Aviation Organization. 2021. Machine Readable Travel Documents – Part 11: Security Mechanisms for MRTDs (Eigth Edition).

[8] International Organization for Standardization. 2018. *Cards and Security Devices for Personal Identification – Contactless Proximity Objects – Part 3: Initialization and Anticollision.* ISO/IEC Standard "14443-3". International Organization for Standardization.

[9] International Organization for Standardization. 2018. *Cards and Security Devices for Personal Identification – Contactless Proximity Objects – Part 4: Transmission Protocol.* ISO/IEC Standard "14443-4". International Organization for Standardization.

[10] International Organization for Standardization. 2020. Identification Cards – Integrated Circuit Cards – Part 4: Organization, Security and Commands for Interchange.

[11] Malte Isberner, Falk Howar, and Bernhard Steffen. 2014. The TTT Algorithm: A Redundancy-Free Approach to Active Automata Learning. In *Runtime Verification (Lecture Notes in Computer Science)*, Borzoo Bonakdarpour and Scott A. Smolka (Eds.). Springer International Publishing, Cham, 307–322. https://doi.org/10.1007/978-3-319-11164-3_26

[12] Malte Isberner, Falk Howar, and Bernhard Steffen. 2015. The Open-Source LearnLib. In *Computer Aided Verification (Lecture Notes in Computer Science)*, Daniel Kroening and Corina S. Păsăreanu (Eds.). Springer International Publishing, Cham, 487–495. https://doi.org/10.1007/978-3-319-21690-4_32

[13] Michael J. Kearns and Umesh Vazirani. 1994. *An Introduction to Computational Learning Theory.* MIT Press.

[14] Stefan Marksteiner, Marjan Sirjani, and Mikael Sjödin. 2023. Using Automata Learning for Compliance Evaluation of Communication Protocols on an NFC Handshake Example. In *Engineering of Computer-Based Systems* (Cham) *(Lecture Notes in Computer Science, Vol. 14390)*, Jan Kofroˇ n, Tiziana Margaria, and Cristina Seceleanu (Eds.). Springer Nature Switzerland, 170–190. https://doi.org/10.1007/978-3-031-49252-5_13

[15] Maik Merten, Falk Howar, Bernhard Steffen, and Tiziana Margaria. 2012. Automata Learning with On-the-Fly Direct Hypothesis Construction. In *Leveraging Applications of Formal Methods, Verification, and Validation*, Reiner Hähnle, Jens Knoop, Tiziana Margaria, Dietmar Schreiner, and Bernhard Steffen (Eds.). Vol. 336. Springer Berlin Heidelberg, Berlin, Heidelberg, 248–260. https://doi.org/10.1007/978-3-642-34781-8_19

[16] International Civil Aviation Organization. 2021. Machine Readable Travel Documents – Part 9: Deployment of Biometric Identification and Electronic Storage of Data in MRTDs (Eigth Edition).

[17] R. L. Rivest and R. E. Schapire. 1989. Inference of Finite Automata Using Homing Sequences. In *Proceedings of the Twenty-First Annual ACM Symposium on Theory of Computing (STOC '89).* Association for Computing Machinery, New York, NY, USA, 411–420. https://doi.org/10.1145/73007.73047

[18] Wouter Smeenk, Joshua Moerman, Frits Vaandrager, and David N. Jansen. 2015. Applying Automata Learning to Embedded Control Software. In *Formal Methods and Software Engineering*, Michael Butler, Sylvain Conchon, and Fatiha Zaïdi (Eds.). Springer International Publishing, Cham, 67–83.

[19] Martin Tappler, Bernhard K. Aichernig, and Roderick Bloem. 2017. Model-Based Testing IoT Communication via Active Automata Learning. In *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST).* 276–287. https://doi.org/10.1109/ICST.2017.32