Improving Test Conformance of Smart Cards versus EMV-Specification by Using on the Fly Temporal Property Verification
Germain Jolly, Sylvain Vernois, Jean-Luc Lambert

To cite this version:
Germain Jolly, Sylvain Vernois, Jean-Luc Lambert. Improving Test Conformance of Smart Cards versus EMV-Specification by Using on the Fly Temporal Property Verification. International Conference on Security in Computer Networks and Distributed Systems, Mar 2014, Thiruvananthapuram, India. 10.1007/978-3-642-54525-2_17. hal-01149208

HAL Id: hal-01149208
https://hal.archives-ouvertes.fr/hal-01149208
Submitted on 6 May 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Improving Test Conformance of Smart Cards versus EMV-Specification by using on the Fly Temporal Property Verification

Germain Jolly, Sylvain Vernois and Jean-Luc Lambert

Normandie Univ, UNICAEN, GREYC, ENSICAEN, CNRS
6, Boulevard Marchal Juin - F-14050 CAEN cedex - FRANCE
(germain.jolly,sylvain.vernois)@ensicaen.fr, jean-luc.lambert@unicaen.fr

Abstract. Electronic payment transactions using smart card are based on the Europay Mastercard Visa (EMV) specifications. This standard appeared in 1995 in order to ensure security and global interoperability between EMV-compliant smart cards and EMV-compliant payment terminals throughout the world. Another purpose of EMV specifications is to permit a secure control of offline credit card transaction approvals. This paper will expose a way to improve verification and validation of the payment application stored in the chip of the smart card based on temporal property verification. In fact, each issuer (e.g., MasterCard) defines its own EMV-compliant specification, allowing different implementation cases and possible errors and we discuss about a method to detect anomalies to avoid smart card vulnerabilities. The properties will be designed in conformance with EMV-specification but our goal is not to formally prove them. We consider implementations through a black-box testing approach, therefore we cannot prove the properties as we don’t have access to the source code. However, we can observe the command/response exchanges and detect, on the fly, when an expected property is violated.

Keywords: Payment, EMV, Smart Card, Evaluation, Temporal Property.

1 Introduction

According to EMVco, there were 1.55 billion EMV [1] compliant chip-based payment cards in use worldwide in 2012 and it’s still growing. The mass deployment amplifies the security risks and need for the manufacturers to have a head start over the capabilities of attackers. Once created and before being sold, the cards must be certified by a certification authority. This authority will appoint centers specialized in the analysis of vulnerabilities to check if the security schemes are respected. The cards must be verified in order to be validated to be used in everyday life. Regarding the validation of payment applications (e.g., MasterCard applications [2]), we can generate automatically a large number of test cases, for example with fuzzing technics [3]. The problem is that it is difficult for a campaign of intensive testing to trace the root reason of a malfunction...
as steps triggering the error can be generated before the detection of the malfunction. We propose to detect the violation of properties. Properties will be defined in order to understand the cause of a malfunction during the transaction between the terminal and the smart card.

First, the background and limits of evaluation methods will be studied. The second part deals with the main idea and the architecture of a tool created for this study. In the third part, we will talk about our contribution on verification and validation methods with property definition. Finally, the tool realized using the framework WSCT [4] will be exposed.

2 Background

2.1 Communication between the terminal and the smart card

An electronic transaction may be divided into several transactions which are each a coherent set of information exchanged through a network. Transferred data from the terminal to the smart card is called Command APDU (Application Protocol Data Unit) and received data by the terminal from the smart card is called Response APDU. For each command emitted by the terminal, the card sends back one response and acts as a slave in the dialog. It is defined in ISO/IEC 7816 standard [5] and exposed on the figure 1.

![Command APDU](image1.png)

**Fig. 1.** A pair command/response APDU

On the figure 2, we can see the structure of a Command APDU and a Response APDU according to ISO/IEC 7816. CLA indicates the type of the command, INS the specific command, P1 and P2 are parameters for the command, LC indicates the length of the UDC which is optional data and LE indicates the length of the expected data. This expected data, only contained in the response if LE is in the command, is the UDR. Finally, SW1 and SW2 are mandatory in the response and are the command processing status.

![Composition of a Command APDU and a Response APDU](image2.png)

**Fig. 2.** Composition of a Command APDU and a Response APDU
2.2 Card specifications : M/CHIP

In our case, the MasterCard M/CHIP specification [2], an EMV-compliant specification for Mastercard smart cards, is studied. The application contained on the chip can take several states (idle, selected, ...). The evolution of the state of the application is allowed by sending Command APDU (select, get data, GPO, ...) and receiving Response APDU (9000, 6283, ...). A machine state is given by Mastercard to illustrate the M/CHIP application. The possible application states are:

- idle : Application is not currently selected
- selected : Application is selected
- initiated : Transaction is initiated
- online : Application expects a connection with the issuer
- script : Application is ready to accept a script command

The payment application can evolve only by receiving and responding to a series of pairs command/response, this is the concept of application's state. We can't only consider the acceptance of a single command but the response taking into account its current state and then its past evolution.

2.3 Limits of evaluation methods

The smart cards must be certified before being issued by a financial institution. Many evaluation methods exist to help verify and validate smart cards, e.g., fuzzing allows to generate automatically a large number of test cases [3]. However, these methods show a lack of visibility. We can only know if a smart card is correct. We would like to know where is the error of implementation. We need to know how to detect and repair an error with more visibility. Others methods are knowns on Java Card applets like the use of pre- and postconditions is seen in [6] or [7]. But this method is a white box method. In our case, we would like to study marketed smart cards in addition of modified smart cards. We consider payment application, i.e., java card applets through a black box testing approach.

3 A tool to observe the payment transaction

To obtain more information about an error, we can use a temporal analysis, i.e., check that sequences of Commands/Responses APDU during the transaction are correct. This method is often used in the field of electronics. Assertion Based Design [8] allows to verify properties, relationships or sequences on electronics systems. A clock signal is used to know when the system is evolving and when the properties must be verified. In this context, we will do a temporal analysis to improve the evaluation of a smart card.
3.1 WSCT framework

WSCT [9] is a framework developed by S. Vernois et al. for several years. It is written in C# and allows to work with smart cards, e.g. for exploration and finding fault on smart cards [4]. The two main purposes of this tool are to provide:

- an API object-oriented to access a smart card reader.
- an evolutive GUI with creation of plugins to manipulate smart cards.

3.2 Temporal analysis

The main purpose is to improve the known evaluation method by creating an independent module able to observe the system behaviour and to detect when a property is violated. As the state of the application can change only by sending and receiving command/reponse APDU, we can detect improper behavior by observing the input and output of the smart card. These two events are legitimate candidates to define the clock timer that will be used to launch the verification of properties, similarly to a clock signal in the field of semi-conductor with the Assertion Based Design [8].

Using this clock, we can define assertions, properties, sequences on systems. The signals A, B and C can be associated to define a sequence, i.e., \( S : A \text{ and } B \text{ and } C \). This sequence will be true only on the second clock cycle. On the figure 3, you can see the illustration of a clock and its association safety property and signals.

![Clock defined by Command APDU and Response APDU](image)

**Fig. 3.** Clock defined by Command APDU and Response APDU

We observe the Commands and Responses between the chip and the terminal. Then we may observe a property by seeing Commands and Responses...
like signals. Accessing data that normally doesn’t exist on the card application would be a simple property. An other property, a temporal one, could check the refusal of a command, e.g., a Get Processing Option command that initiates the transaction within the card after acceptance of a command that would have normally put the application in a state normally able to accept it. Such error of implementation has already been discovered and are difficult to diagnose.

3.3 Objective

We will simulate the transaction, which can possibly be a fuzzing transaction or a nominal transaction, using the WSCT framework. Previous studies have been done on fuzzing using WSCT. We could use this work and add our tool to improve the results of this work by a higher level of visibility. The goal is not to replace existing work on validation but to improve it. The smart card is connected to WSCT, which can act as a terminal sending preprocessed commands to the card. The observation tool, which purpose is to do the temporal analysis previously presented on smart card applications, is a WSCT plugin made of three modules: an observer to examine the sending and reception of commands and responses, a detector to identify the sequences and a property detector in order to inform the customer, terminal or other entity when a sequence is seen and then a property is violated. These sequences must satisfy some predefined properties and each violated property means there is an incorrect behavior. Figure 4 illustrates how the tool is linked to the transmitted data during the transaction.

![Diagram of the observer and the associated interactions](image)

**Fig. 4.** Diagram of the observer and the associated interactions

The main purpose is to be sure of the validation of a payment application by detecting errors of implementation. We must see it as an additional level of validation to improve the confidence of the smart card. We want to emphasize
the independence of the tool in comparison with communication between the terminal and the card but also the modularity concerning the properties. Several libraries of properties can be created for different smart card applications. And we could easily add or delete properties for the current analysis in order to verify specific part of the implementation or a specific application.

4 Contribution

4.1 Property definition

An improved clock can be defined on the figure 5. By fixing an origin point (0), we can study evolution of the smart card, more precisely the payment application, by studing series of $n$ pairs of command/response. Finally, a property is a local theoretical evolution of the payment application. We can define two kinds of properties: simple properties (only on one pair command/response) and temporal properties (defined with several pairs). The commands match the rising clock cycles. The properties are designed in conformance with EMV-specification. The only thing we can do is to observe the command/response exchanges and detect, on the fly, when an expected property is violated. Indeed, we are not suppose to know the source code.

![Fig. 5. The clock defined during APDU communication](image)

Technically, a property defines a local behavior, i.e., that under certain predicates at a given moment, the smart card must have a specific behavior. As properties are defined as associations of commands and responses, we can describe all kind of features because we can observe the behavior of the application by observing the APDU communication. The commands are associated with even numbers and the responses with odd numbers. Therefore, we can only define properties with CLA, INS, P1, P2, LC, UDC and LE on even clock cycles and with UDR, SW1 and SW2 on odd clock cycles. The value of some fields involve a reaction from the smart card and consequently the value of others fields. The property must be checked to ensure that the local behavior of the smart card is correct. $P_0$ shows the most natural structure of a property, as an implication. By logical equivalence, we can use a second form to define $P_0$. Indeed, we have $A \implies B$ equivalent to $B \lor \neg(A)$.
$P_0 : (\text{field}(0) \text{ and } \ldots \text{ and field}(i)) \Rightarrow (\text{field}(1) \text{ and } \ldots \text{ and field}(j))$

$P_0 : (\text{field}(1) \text{ and } \ldots \text{ and field}(j)) \text{ or } (\text{field}(0) \text{ and } \ldots \text{ and field}(i))$

with $0 < i < j$

4.2 Simple properties and predicates on one time clock only

We expose two simple properties in order to illustrate our work. Indeed, a simple thing is to verify the integrity of the command and response. The property $P_1$ check the correctness of the response. Is the sent data by the smart card correct according the received command? Here, we are verifying that the length of the expected data (UDR) in the response is LE, given in the previous command. In this case, only one time clock is checked (we are studying the 0 and 1 time clock).

Response length correct :

$P_1 : (\text{UDR}(1) \neq 0) \text{ and } (\text{UDR}(1).\text{length} = \text{LE}(0)) \text{ or } (\text{LE}(0) \neq 0)$

This figure 1 illustrates a simple property to show the link between the sent and received data and the property definition. It’s the generalization of the two previous properties. The command (defined by an association of fields with specific values) implies the response (defined by an associated of fields with specific values). We can also verify the value of the fields contained in the command plus the value of the fields contained in the response.

4.3 Complexes properties defined on n times clock

We can check series of pairs command/response APDU too. According to one send command, the property $P_2$ allows to detect a replay. In fact, if every bytes of the second command are the same than the first command, this property will be false. For each two command/response pairs, we have to check the property is correct (at least one of the fields of the second command is different) to avoid replay during the transaction.

No command replay detected :

$P_2 : (\text{CLA}(2) \neq \text{CLA}(0)) \text{ or } (\text{INS}(2) \neq \text{INS}(0)) \text{ or } (\text{P1}(2) \neq \text{P1}(0)) \text{ or } (\text{P2}(2) \neq \text{P2}(0)) \text{ or } (\text{LC}(2) \neq \text{LC}(0)) \text{ or } (\text{UDC}(2) \neq \text{UDC}(0)) \text{ or } (\text{LE}(2) \neq \text{LE}(0))$

The last exposed property $P_3$ permit to detect an error of implementation in the application. We can see the theorical behaviour of the smart card notably between the "selected" state and the "initiated" state using only the GPO command (‘A8’ is the INS for Get Processing Options Command) in the MasterCard
specification [2]. We could generalize it to detect any wrong behavior between these two states.

Good behavior with three GPO commands:

\[
P_3 : ((SW1(1) = 90) \text{ and } (SW2(1) = 00) \text{ and } (SW1(3) \neq 90) \text{ and } (SW2(3) \neq 00)) \text{ or } (INS(0) = A8) \text{ and } (INS(2) = A8) \text{ and } (INS(4) = A8)
\]

The figure 5 illustrates the \( P_3 \) property that involves several pairs command/response, so three clock cycles. The commands associated to the number 0, 2 and 4 of the clock timer are defined by an association of fields with specific values and implies the responses associated to the number 1, 3 and 5 defined by an associated of fields with specific values. In fact, the last response is due to the previous commands. The property is finally a set of values to be checked and is true if the application is correct.

5 The observation tool

5.1 WSCT and the observation plugin

In order to do the tool presented in the figure 4, we have created two plugins on WSCT. We have grouped the three modules: the Observer, the Property Detector and the Property Observer into one plugin. In fact, its purpose is to take as inputs the APDU commands and responses and compute if the properties are checked or not, in order to verify the smart card with a higher level of appreciation. The plugin called Transaction do the communication with the smart card.

On the figure 6, you can see how we have grouped the modules and linked them with WSCT. Two plugins have been created: one, called Transaction, allows to launch any type of terminal (fuzzing, nominal, ...) [10] and the other, called Observer, is the tool to observe properties.

![Fig. 6. WSCT, its perimeter and the first version of the tool](image)

From APDU communication, the Observer plugin can construct all the properties to be verified. If one property is violated, the user must be warned. We can
observe independently of the transaction processing the data exchanged between a smart card and the transaction application. Violation of simple properties can also be detected, e.g., detecting wrong command or replay.

5.2 Verification of properties using this tool

The figure 7 describes the algorithm to verify properties. One property is an object using variables containing the value of commands and responses APDU captured at a given moment on a discrete and bounded time-line (0, 1, 2...n). We instantiate the variable of the properties by capturing APDU communication on the fly. When all the variables of a property have been instantiated, we can verify their validity and interrupt the terminal or send a signal to the user if there is a violation. Then all the properties which verification is done (all their variables were instantiated) are killed and new ones are created that will check the same properties in the forthcoming instants.

![Fig. 7. Verification process of properties](image)

5.3 Advantages of our verification method

Finally, this method shows four main advantages:

- Black-box verification: We need neither the source code of the smart card neither the source code of the terminal;
– Generic verification: This method can be used on other kinds of smart card application;
– On the fly verification: This method is independant of the transaction process by only capturing the communication;
– Modular verification: A library of properties or just a specific property can be verified.

6 Conclusion

In this paper, we have exposed an additional method to improve software evaluation of smart cards. We can easily isolate a property on the fly to obtain more information about a detected error. This study, called temporal analysis, allows us to know what really happened and especially when it happened through the use of temporal properties. Through a black box method, we are able to determine sequences of data exchanged between the smart card and the terminal which permit us to detect wrong behavior of the payment application. Finally, we can strengthen the validation of smart cards by using our method over the methods already used.

This work will be continued and we will focus on several main topics. First, the use of temporal properties for software evaluation must be validated. To evaluate our work, we will use validated smart cards and modified smart cards containing errors. This method will be used on several types of applications, including M/CHIP applications. The next step of this work is an automatic generation of properties to verify on a specific application. The tool should allow to learn properties all along the execution too in order to become a flexible and evolving tool.

References

1. EMV Integrated Circuit Card Specifications for Payment Systems, version 4.3 EMVco, 2011
3. Un framework de fuzzing pour cartes a puce: application aux protocoles EMV, J. Lancia, 2011
5. ISO/IEC 7816, International Organization for Standardization and the International Electrotechnical Commission
8. Assertion-Based Design, Harry D. Foster, Adam C. Krohnik, David J. Lacey, 2010
10. Analyse de la s´ecurit´e de transactions `a puce avec le framework WinSCard Tools, Benoˆıt Vibert, Vincent Alimi, Sylvain Vernois, 2012