Emulator of Contactless Smart Cards in FPGA

Stanislav Jeřábek, Jiří Buček, Jan Schmidt and Martin Novotný
Faculty of Information Technology
Czech Technical University in Prague
Czech Republic, Prague
{jerabst1, bucekj, Schmidt, novotnym}@fit.cvut.cz

Abstract—**This paper describes implementation of contactless smart card emulator compliant with ISO/IEC 14443 in *Field Programmable Gate Array* (FPGA). Systems using contactless smart cards are widely used and some of these systems are not secured properly. For example in many such systems smart card Unique Identifier (UID) is used as the only one authentication mean. As the UID is not encrypted and is read from the card in plain, it is easy to make a copy of the smart card and use the clone as the original card. In this work we describe emulator of a smart card implemented in FPGA which is able to spoof some genuine smart card. Emulator described in this work emulates protocol described in ISO/IEC 14443 standard, which in detail describes all aspects of RFID smart cards (from physical attributes of both – cards and readers – to communication by digital signals). The emulator is able to come through the whole *card selection* process and to spoof the real smart card with given UID. Moreover emulator can be selected also for higher application layer protocol communication. If we know the proprietary application layer protocol, emulator is able to spoof communication on this protocol with data recorded in it. This functionality was successfully tested on systems used at Czech Technical University in Prague, where the weak implementation of UID as the only one authentication mean without any proprietary protocol is used. Emulator is responding faster than most of other existing smart card emulators thanks to high efficient implementation in hardware.**

Keywords-FPGA; emulation; contactless Smart Card; ISO/IEC 14443

I. INTRODUCTION

Smart cards are widely used in a lot of different systems. Smart cards themselves usually offer some security thanks to internal data blocks and usage of different ciphers during communication on higher protocol layers. However, there are some systems, where unsecure implementation of smart card technology is used (for example systems without any other authentication mean than UID of a smart card). The UID is therefore public and sent as plaintext. Topic of smart card security is well researched and several different weaknesses of smart cards security have been discovered. There are also some devices, which are able to emulate smart cards.

Nevertheless, the popularity of smart cards is still rising and the contactless smart cards are still more favorite in comparison with the other smart cards. The most well-known use of smart cards are credit and debit cards for banking and Subscriber Identity Module (SIM) cards in telecommunications. Smart cards can be also used in a lot of proprietary systems in many ways, where contactless smart cards are used more than other smart cards [1].

Smart cards can also be used in a lot of commercial applications such as banking applications, payments of parking and traffic fares and a lot of others [2].

II. PREVIOUS WORK

There are some existing projects implementing emulator of contactless smart cards. Particular examples are *Proxmark3* [3], *Simple NFC* [4] or *ChameleonMini* [5]. All these projects support the ISO/IEC 14443 standard [6]. Complexity of these projects differs but the most important common attribute is that some microprocessor is used for management of the whole system.

- *Proxmark3* is able to emulate smart card, smart card reader and also sniff communication of real smart card and reader. All of these can be done with use of low or higher frequencies.
- *Simple NFC* is able to emulate Near Field Communication (NFC) tags according to parts 1-3 of ISO/IEC 14443 standard [6].
- *ChameleonMini* supports also some higher protocol layers and therefore supports higher functionalities of several specific smart card types. The device can be also used for executing of some attacks.

The main contribution of this paper consists in continuing development of an existing smart card emulator created in a master thesis [7]. It consists of board *Nexys3* with *Xilinx Spartan-6* FPGA [8] and *PN532 Breakout board* [9] in *Virtual card mode* as radio interface. When *PN532 Breakout board* works in *Virtual card mode* as rectifier and uses hysteresis to make digital signal from surrounding RF field, which is shown in Fig. 1. It also allows load modulation of RF field as also shown in Fig. 1. That is used for communication from card to reader. The emulator was able to spoof any smart card UID according to part 1-3 of ISO/IEC 14443 standard before any upgrades. However, emulator was not able to communicate with application data protocol according to the fourth part of standard [6]. Now the emulator is still under development to be

able to do so. The biggest benefit of hardware implementation (better timing and performance) remains retained.

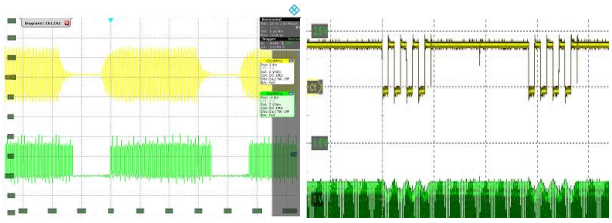

Figure 1. Radio interface (*PN532 Breakout board*) processing: Left: RF field(up) and *SigOut* signal (down) after processing. Right: Input *SigIn* signal (up) load modulating RF field (down)

III. EMULATOR DESIGN AND IMPLEMENTATION

We still use board *Nexys3* with FPGA *Xilinx Spartan-6* family [8] as target platform for the emulator. There is also still used *PN532 Breakout board* as radio interface [9]. The new part of system will be used a microchip for processing of transactions on higher application layer. So the emulator of smart card itself remains in FPGA and is supported with another device as radio interface and a microchip as transaction controller. FPGA and radio interface processes RF traces, decodes them as lower layer transactions and after that these transaction data are sent as higher layer data into microcontroller for other processing. Usage of microcontroller only for application data processing and FPGA for all decoding and RF communication offers more efficient implementation of the smart card emulator. Although there remain a lot of unused hardware resources in actually used FPGA (the attribute with the biggest usage is *Slices used* with value about 14 %), we are going to use some microcontroller for application layer data processing. There is a proprietary protocol in application layer and it would be unconvinient to make changes in VHDL code everytime this proprietary protocol is changed. This also gives possibility to other users to implement their own proprietary protocol in any microcontroller and connect it with FPGA smart card emulator.

A. Design of Extensions

Microcontroller processing any commands on higher protocol layer defined by a user is connected via serial protocol. Originally RS-232 was intended as the serial protocol but its throughput even with baud rate 115200 bauds/s turned out as insufficient. Data were being sent to slow for responding so fast as ISO/IEC 14443 standard allows [6]. SPI was after that chosen as communication protocol between FPGA and microcontroller. While the emulator is successfully selected according to the fourth part of the standard [6], it works in *bytemode* and sends received bytes to microcontroller for processing and after that sends response back to card reader. All these user-defined protocol commands are absolutely independent of the emulator itself. Another emulator upgrade consists in another serial (this time RS-232) protocol extension which provides comfortable run-time reconfiguration of emulated smart card. It was necessary to compile new design for configuration change because the configuration data were part of the VHDL code.

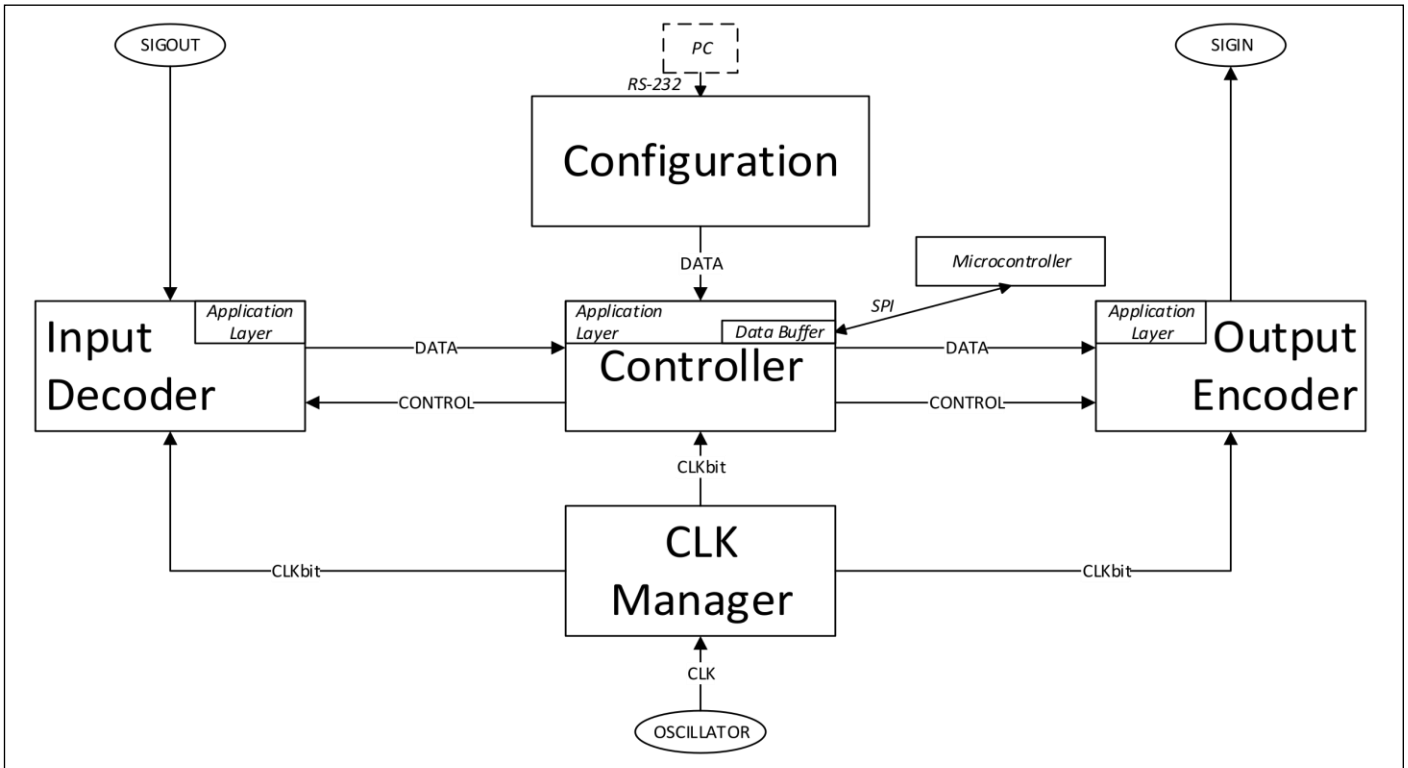

Figure 2. Simplified design of implemented system. Italic written parts (*PC, Application Layer, Data Buffer* and *Microcontroller*) are extensions described in this paper. *PC* stands for any device used for configuration.

B. Implementation

Simplified design of implementation in FPGA and connection with some other devices is shown in Fig. 2. Italic written parts are extensions described in this paper. Some extensions are brand new parts, but some changes (extensions) had to be made in existing entities. Part called as *PC* is any device used for configuration, not any part of the emulator itself.

There are three different frequencies used when emulator decodes data form physical layer. The carrier frequency (13.56 MHz) is used by Radio Frequency (RF) field, which provides power for smart card and also serves for communication [6]. Card reader can start transmitting almost anytime (there is only restriction of start early after previous communication was finished). There are also two different frequencies derived from carrier frequency: bitrate and subcarrier. Both of them have to be synchronous with carrier frequency. Moreover FPGA has RS-232 interface for run-time smart card reconfiguration and SPI protocol interface for communication with microcontroller processing data on application layer.

For designing connection with microcontroller thus enhancing the whole design with application data processing, some decisions about parameters had to be made. First, for purpose of system, some proprietary application layer protocol has to be designed. Because of this protocol is proprietary, it can't be used for spoofing some genuine smart card unless we know some information about genuine protocol and also some data on genuine smart card if they are used for security. One of these security measurements can be for example serial number of transaction, which is used as protection against copying some genuine smart card with credit.

Another crucial decision is about length of data blocks. There are transmitted some data blocks containing proprietary data on application layer. According to the standard the data blocks can be from 16 to 256 bytes long [6]. To send these data blocks from smart card emulator itself (FPGA) into microcontroller there have to be some data queues Length of these queues depends on maximum data block length. For our emulator we can use rather short data blocks which give us opportunity to save some hardware resources with minimal decrease of data throughput. Total throughput is by overhead coming from more often transactions influenced only a little, thanks to very efficient hardware implementation of emulator. When using FPGA with 100 MHz clock source and actual design can be responses on lower layer (without application processing by microcontroller, e.g. card selection) transmitted back to card reader in 40 ns after incoming data are received (4 clock cycles with 9,6 ns critical datapath length). Actual time is much higher (circa 86 μs) because of requirements of the standard [6].

C. Tests

Almost all components were verified with behavioral simulation. Parts of system that implement first three parts of standard [6] (without proprietary protocol) were then successfully validated with a smart card reader. It means that emulator was successfully selected with its UID and communication was established.

When implementing and testing the emulator according to ISO/IEC 14443 standard several unexpected issues had to be solved. All of them were caused by using recommended zero values by original standard in bits reserved for future use [6]. Since the standard was released, a lot of other standards for different smart card subsets were released while they are using some values reserved for future use in the initial standard [10, 11]. There smart card subtypes has different improvements and protocol additions such as different data block structures or built-in encryption but they have in common everything in initial ISO/IEC 14443 standard [6].

IV. Implementation Results

We have successfully implemented contactless smart card emulator using *Nexys3* as emulator platform and *PN532 Breakout board* as radio interface. Also extension for run-time reconfiguration of emulated smart card with RS-232 protocol has been implemented. Now we are working on extension with a microcontroller for processing proprietary application layer data protocol.

Figure 3. Emulator without microcontroller for application data processing powered wih a power bank.

The design is therefore suitable also for much smaller (count of slices and physically too) FPGAs. The whole design will be a little bigger because of extensions. Run-time reconfiguration possibility will enlarge design with one RS-232 module and application layer extension will enlarge design with SPI communication module and also main controller will be a little bigger because of new smart card internal states according to the standard. The whole system without microcontroller for application data processing powered by a power bank is shown in Fig. 3.

The emulator implements all four parts of ISO/IEC 14443 standard for smart cards type A. Protocol for smart cards of type B is not implemented [6].

The emulator allows to change on run-time smart card attributes such as UID (unique identifier), Select Acknowledgment (SAK) and Answer to Request (ATQA). The emulator is able to process application layer commands defined

by user with connected microcontroller. These commands are received via RF field by emulator and as byte arrays they are sent to microcontroller for processing itself. Design supports block containing 32 bytes of data, which is helpful for saving hardware resources and still reducing total throughput by transaction overhead only minimally.

V. CONCLUSION

The emulator is able to spoof any smart card in weak implemented system by spoofing its UID, SAK and ATQA. The device also emulates smart card with data blocks communicating on higher application layer protocol according to the standard. If we had some information about content of real card and commands used for communication, we would be able to spoof card and its data in real system without smart card key or cipher security.

Smart card configuration is now loaded into emulator through SPI protocol connection and can be changed on run-time.

Implementation almost exclusively in hardware (FPGA) offers lower latency and higher performance in comparison with software-based emulators. Usage of microcontroller for application layer data processing has minimal impact.

ACKNOWLEDGMENT

This work was partially supported by the grant GA16-05179S of the Czech Grant Agency, "Fault Tolerant and Attack-Resistant Architectures Based on Programmable Devices: Research of Interplay and Common Features" (2016-2018). This research has been in part supported by CTU grant SGS17/213/OHK3/3T/18. I would like to thank to Denis Titov for his work on this project in his Bachelor thesis.

REFERENCES

[1] Chhabra, T.; Chindaphorn, P. Smart Cards. [online], 2004, Santa Clara University, Department of Computer Engineering, [Cited 2017-02-02]. Available from: http://www.cse.scu.edu/~jholliday/COEN150Sp04/projects/Smart%20Cards.doc

[2] *Unknown.* Uses of Smart Cards. [online], 1997, Massachusetts Institute of Technology, [Cited 2017-02-02]. Available from: http://web.mit.edu/ecom/Spring1997/gr12/2USES.HTM

[3] Westhues, J. A Test Instrument for HF/LF RFID. [online], 2009, [Cited 2017-02-15]. Available from: http://www.cq.cx/proxmark3.pl

[4] Kruse, N. Simple NFC. [online], 2013, [Cited 2017-02-15]. Available from: http://blog.nonan.net/2013/11/simple-nfc.html

[5] Kasper, T. Chameleon Project. [online], [Cited 2017-02-15]. Available from: https://github.com/emsec/ChameleonMini/wiki

[6] International Organization for Standardization. ISO/IEC 14443 - Identfication cards – Contactless integrated circuit cards - Proximity cards. 2001.

[7] S. Jeřábek, "Emulátor bezkontaktní čipové karty v FPGA," Diplomová Práce. Praha: Czech Technical University, Faculty of Information Technology, 2016.

[8] Digilent. Nexys3 Board Reference Manual. [Revised 2013-04-03], [Cited 2017-03-05]. Available from: https://www.xilinx.com/support/documentation/university/XUP%20Boards/XUPNexys3/documentation/Nexys3_rm.pdf

[9] NXP Semiconductors. UM0701-02 - PN532 User Manual. 2007, [Cited 2017-03-05]. Available from: http://cache.nxp.com/documents/user_manual/141520.pdf?fsrch=1&sr=1&pageNum=1

[10] NXP Semiconductors. AN1303 MIFARE Ultralight as Type 2 Tag. Version 1.5, [Revised 2012-03-05], [Cited 2017-03-05]. Available from: http://cache.nxp.com/documents/application_note/AN1303.pdf?fsrch=1&sr=1&pageNum=1

[11] NXP Semiconductors. AN10833 MIFARE Type Identfication Procedure. Version 3.6, [Revised 2016-07-11], [Cited 2017-03-05]. Available from: http://cache.nxp.com/documents/application_note/AN10833.pdf?fsrch=1&sr=1&pageNum=1