When Organized Crime Applies Academic Results
A Forensic Analysis of an In-Card Listening Device

Houda Ferradi, Rémi Géraud, David Naccache, and Assia Tria

[1] École normale supérieure
Computer Science Department
45 rue d'Ulm, F-75230 Paris CEDEX 05, France
Email: given_name.family_name@ens.fr
[2] CEA-TEC PACA
Centre Microélectronique de Provence
880 Route de Mimet, 13541 Gardanne, France
Email: assia.tria@cea.fr

Abstract. This paper describes the forensic analysis of what the authors believe to be the most sophisticated smart card fraud encountered to date. In 2010, Murdoch *et al.* [7] described a man-in-the-middle attack against EMV cards. [7] demonstrated the attack using a general purpose FPGA board, noting that "*miniaturization is mostly a mechanical challenge, and well within the expertise of criminal gangs*". This indeed happened in 2011, when about 40 sophisticated card forgeries surfaced in the field.
These forgeries are remarkable in that they embed two chips wired top-to-tail. The first chip is clipped from a genuine stolen card. The second chip plays the role of the man-in-the-middle and communicates directly with the point of sale (PoS) terminal. The entire assembly is embedded in the plastic body of yet another stolen card.
The forensic analysis relied on X-ray chip imaging, side-channel analysis, protocol analysis, and microscopic optical inspections.

1 Introduction

EMV [2–5] (Europay, MasterCard, Visa) is a global standard, currently managed by the public corporation EMVCo, specifying interactions between integrated circuit cards and PoS terminals. The standard also defines exchanges between cards and automatic teller machines (ATMs). Over the recent years, additional payment operators (such as JCB, AmericanExpress, China UnionPay and Discover) endorsed EMV. EMV cards rely on pre-existing physical, link, network, and transport layer protocols such as ISO/IEC 7816 and ISO/IEC 14443.

According to EMVCo's website, by Q4 2014 a third of card present transactions worldwide followed the EMV protocol, and 3.423 billion EMV cards were in circulation.

1.1 Brief Overview of an EMV Transaction

A typical EMV transaction breaks down into three phases: (1) card authentication, (2) cardholder verification and (3) transaction authorization.

During card authentication, the PoS explores the applications supported by the card (*e.g.* credit, debit, loyalty, ATM, etc.).

During cardholder verification, the PoS queries the PIN from the user and transmits it to the card. The card compares the PIN and responds by "yes" (`SW` code[3] `0x9000`) or "no" (`0x63CX`[4]).

Transaction authorization starts by feeding the card with the transaction details T (*e.g.* amount, currency, date, terminal ID, fresh randomness, etc.). The card replies with an authorization request cryptogram (ARQC) based on T. {ARQC, T} is sent to the issuer[5], who replies with

[3] Whenever a command is executed by a card, the card returns two status bytes called `SW1` and `SW2`. These bytes encode a success or a failure cause.
[4] `X` denotes the number of further PIN verifications remaining before lock-up.
[5] For our purposes, the issuer can be thought of as the bank.

an authorization request code (ARC) instructing the PoS how the transaction should proceed. The issuer also sends to the PoS an authorization response cryptogram (ARPC) which is a MAC of {ARQC, ARC}. ARPC is transmitted to the card that responds with a transaction certificate (TC) sent to the issuer to finalize the transaction.

We refer the reader to [7] for a comprehensive diagram illustrating these three phases.

1.2 Murdoch et al.'s Attack

The protocol vulnerability described in [7] is based on the fact that the card does not condition transaction authorization on successful cardholder verification.

Hence the attack consists in having the genuine card execute the first and last protocol phases, while leaving the cardholder verification to a man-in-the-middle device.

To demonstrate this scenario's feasibility, Murdoch *et al.* produced an FPGA-based proof-of-concept, noting that miniaturisation remains a mechanical challenge.

1.3 Fraud in the Field

Fig. 1. The judicial seizure. Personal information such as cardholder name are censored for privacy reasons.

In May 2011, the French's bankers Economic Interest Group (GIE Cartes Bancaires) noted that a dozen EMV cards, stolen in France a few months before, were being used in Belgium. A police investigation was thus triggered.

Because transactions take place at well-defined geographic locations and at well-defined moments in time, intersecting the IMSIs[6] of SIM cards present near the crime scenes immediately revealed the perpetrators' SIM card details. A 25 years old woman was subsequently identified and arrested, while carrying a large number of cigarette packs and scratch games. Such larceny was the fraudsters' main target, as they resold these goods on the black market.

Investigators quickly put a name on most of the gang members. Four were arrested, including the engineer who created the fake cards. Arrests occurred in the French cities of Ezanville, Auchy-les-Mines and Rouvroy. About 25 stolen cards were seized, as well as specialized software and €5000 in cash.

The net loss caused by this fraud is estimated to stand below €600,000, stolen over 7,000 transactions using 40 modified cards.

A forensic investigation was hence ordered by Justice [1].

2 Physical Analysis

2.1 Optical Inspection

The forgery appears as an ISO/IEC 7816 smart card. The forgery's plastic body indicates that the card is a VISA card issued by Caisse d'Épargne (a French bank). The embossed details are: PAN[7] = `4978**********89`; expiry date in 2013[8]; and a cardholder name, hereafter abridged as P.S. The forgery's backside shows a normally looking CVV[9]. Indeed, this PAN corresponds to a Caisse d'Épargne VISA card.

The backside is deformed around the chip area (Figure 2). Such a deformation is typically caused by heating. Heating (around 80°C) allows melting the potting glue to detach the card module.

Fig. 2. Deformation due to heating of the forgery's backside.

The module looks unusual in two ways: (1) it is engraved with the inscription "FUN"; and (2) glue traces clearly show that a foreign module was implanted to replace the `**89` card's original chip (Figure 3).

[6] International Mobile Subscriber Identity.
[7] Permanent Account Number (partially anonymized here).
[8] Precise date removed for privacy reasons.
[9] Card Verification Value.

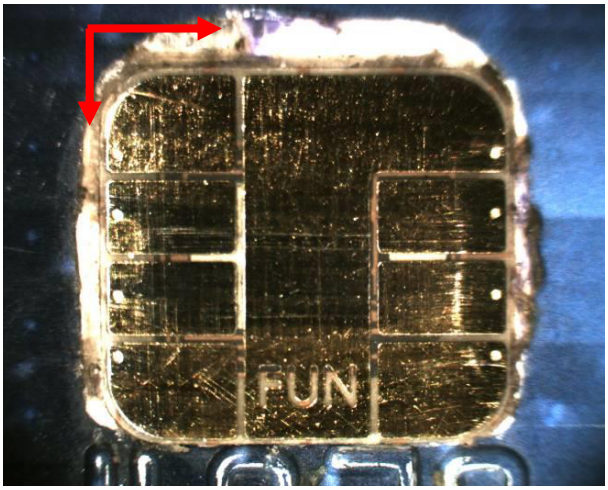

Fig. 3. Forgery's ISO module. Red arrows show glue traces.

The module is slightly thicker than normal, with the chip bulging somewhat through the card, making insertion into a PoS somewhat uneasy but perfectly feasible (Figure 4).

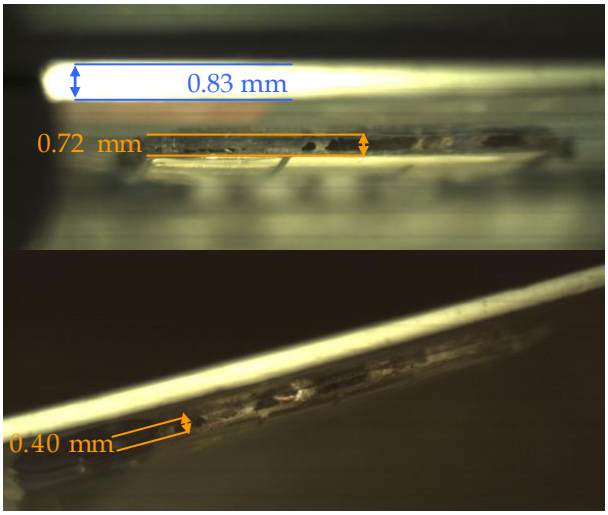

Fig. 4. Side-views of the forgery, showing that it is somewhat thicker than a standard card (0.83 mm). The extra thickness varies from 0.4 mm to 0.7 mm suggesting the existence of several components under the card module, besides the FUN card.

The "FUN" engraving indicates that the module belongs to a FUN card. FUN cards are open cards, widely used for hobbying and prototyping purposes.

The FUN module contains an Atmel AVR AT90S8515 microcontroller and an EEPROM memory AT24Cxx. The AVR has 8 kB of Flash memory and 512 bytes of internal EEPROM, 512 bytes of internal RAM and a few other resources (timer, etc.). The AT24Cxx has varying capacity depending on the exact FUN card model. For a FUNcard5, this capacity is 512 kB (Figure 5).

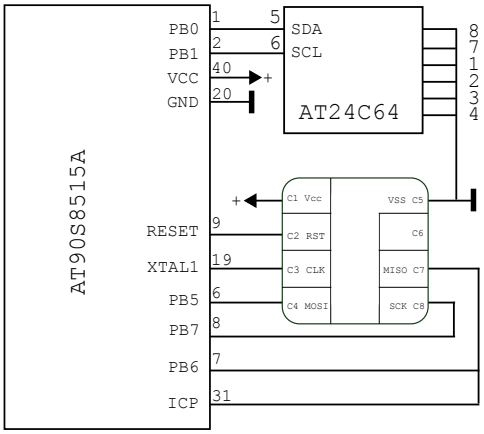

Fig. 5. The FUN card's inner schematics.

2.2 Magnetic Stripe Analysis

The magnetic stripe was read and decoded. The ISO1 and ISO2 tracks perfectly agrees with the embossed information. ISO3 is empty, as is usual for European cards.

2.3 X-Ray Analysis

X-ray analysis was performed using a Y.Cougar Microfocus Xylon imager. Figure 6 shows an unmodified FUN card, while Figure 7 is an X-ray image of the forgery.

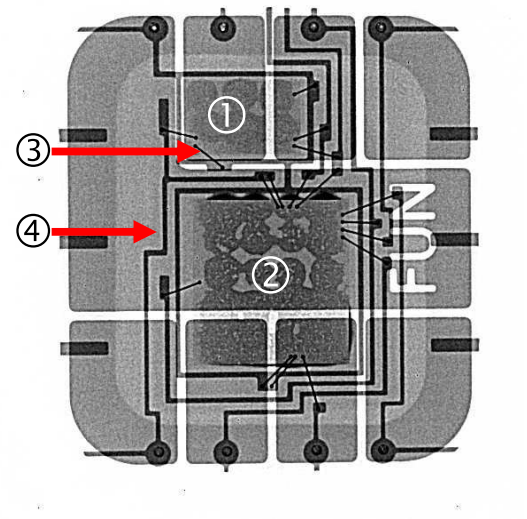

Fig. 6. FUN card X-ray analysis. (1) External memory (AT24C64); (2) Microcontroller (AT90S8515A); (3) Connection wires; (4) Connection grid.

X-ray analysis reveals, using false colours, the different materials composing the forged module (Figure 9). Legitimate connection wires are made of gold, but connections between the FUN card and the stolen chip underneath are made of another metal (copper, as will later appear after opening the forged card). Soldering was made using a classical mixture of silver and tin.

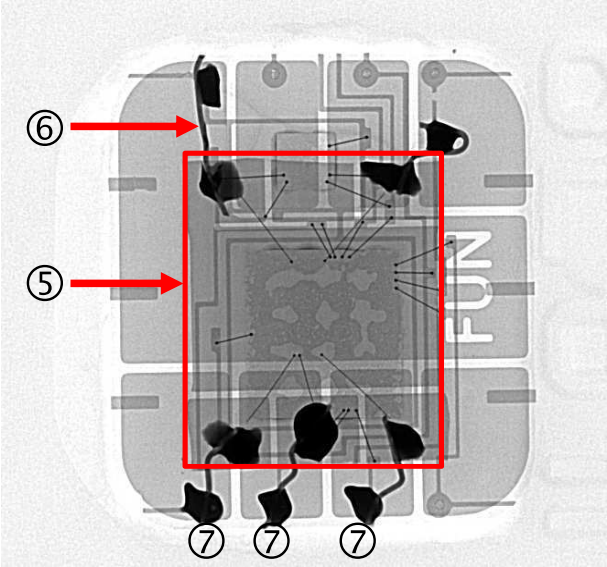

Fig. 7. Forgery X-ray analysis. (5) Stolen card's module; (6) Connection wires added by the fraudster; (7) Weldings by the fraudster (only three are pointed out here).

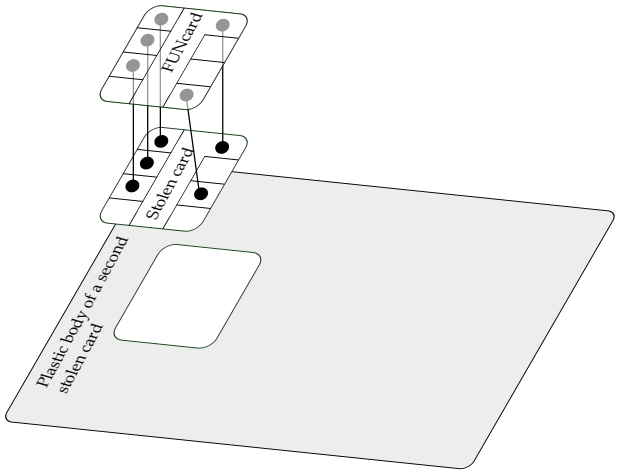

Fig. 8. Forgery structure suggested by Figure 7.

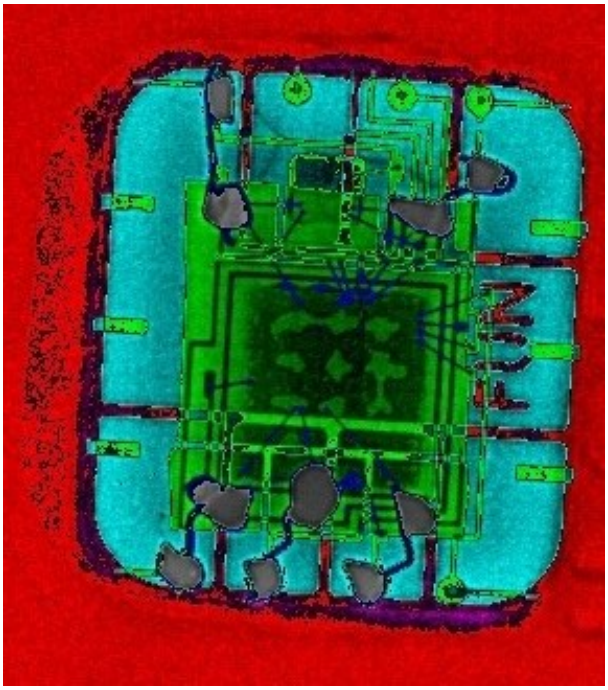

Fig. 9. False colours X-ray image of the forgery. Different colours correspond to different materials. The stolen chip is clearly visible in green.

2.4 Probing non-ISO Contacts

FUN cards are programmed using specialised hardware.

Programming is done *via* the two unstandardized pins `MOSI` and `SCK`.

We tried to use programming hardware to read back the card's contents and reverse-engineer its software.

All reading attempts failed. FUN cards can be protected against reading at flashing time. Clearly, the fraudster enabled this protection.

It is possible to identify the chip using the programming hardware, but this uses writing commands that are invasive and possibly destructive. Therefore such an identification was not attempted.

2.5 ISO/IEC 7816 Compliance

We assumed that the forged card's software was rudimentary and did not fully comply with ISO/IEC 7816. The assumption was that the fraudsters contented themselves with a minimal implementation that works reasonably well under usual conditions. We hence analyzed the forgery's behavior at external clock frequencies close to the most extreme value (5 MHz) allowed by ISO/IEC 7816.

The forgery behaved normally up to 4.90 MHz. At 4.91 MHz, the forgery stopped responding to commands and only returned an ATR (Answer To Reset).

3 Protocol Analysis

The electronic exchanges between the forgery and the PoS were monitored using the `Cardpeek`[10] tool. `Cardpeek` allows monitoring the APDU commands sent to the forgery.

[10] See `http://code.google.com/p/cardpeek/downloads/list`.

This is a read-only operation that does not alter the analyzed card.

When queried, the forgery responded with the following information: PAN = `4561**********79`; expiry date in 2011; and the cardholder name henceforth refered to as H.D. All this information is in blatant contradiction with data embossed on the card (mentioned in Section 2.1).

3.1 Application Selection

Application selection is performed by browsing the PSE[11], as described in [3].

Select 1PAY.SYS.DDF01. Selecting the DDF[12] named `1PAY.SYS.DDF01` succeeded[13].

Browsing the Payment System Directory Records in the Payment System Directory were browsed in-order and revealed the presence of CB[14] and VISA applications.

ReadRecord SFI[15] 1 record #1[16]: this SFI contains:

- Application AID `A0 00 00 00 42 10 10`
- Label: CB
- Priority: 1

ReadRecord SFI 1 record #2: this SFI contains:

- Application AID: `A0 00 00 00 03 10 10`
- Label: Visa DEBIT
- Priority: 2

Attempting ReadRecord SFI 1 record #3 returns a status word equal to `0x6A83`, *i.e.* "record not found". All applications have thus been found.

Select "VISA Debit" `Cardpeek` used the previously discovered AID to select the VISA Debit application[17].

3.2 Transaction Initialization

GetProcessingOptions (GPO) Next, we retrieved the card's processing options[18]. This data contains the AIP (Application Interchange Profile) and the AFL (Application File Locator) as defined in [5, Chapter 10.2]. The card claims that it supports:

- static authentication (SDA);
- dynamic authentication (DDA);
- cardholder verification;
- and external authentication.

The card furthermore requests risk management by the PoS.

AFL consists in a list of 4-byte blocks describing which records should be read. In our case, the following blocks were received:

[11] Payment System Environment.
[12] Directory Definition File.
[13] Command: `00 A4 04 00 14`.
[14] Carte Bancaire.
[15] Short File Identifier.
[16] Command: `00 B2 xx 0C Le`, where `xx` is incremented as records are being read.
[17] Command: `00 A4 04 00 07`.
[18] Command: `80 A8 00 00 02` followed by a GetResponse command: `00 C0 00 00 20`.

– SFI #1, of record 01 to 01. No record of this SFI is used for "disconnected" mode data authentication.
– SFI #2, of record 01 to 02. Record #1 of this SFI is used for "disconnected" mode data authentication.
– SFI #3, of record 01 to 04. Record #1 of this SFI is used for "disconnected" mode data authentication.

SFI Records Having read the GPO data, the reader can access the SFI records.

ReadRecord SFI 2 record #1 (used for "disconnected" mode data authentication) contained the following VISA application information:

– Application creation and expiry date: between a date in 2009 and a date in 2011 (omitted here for privacy reasons).
– The card's PAN: `4561**********79`.

The Bank Identification Number of this PAN corresponds to a HSBC VISA card (4561), which is inconsistent with the information embossed on the card.

ReadRecord SFI 2 record #2 of the VISA application provided the following information:

– The list of objects to be included upon the first GenerateAC (CDOL1) (tags list + lengths)
– The list of objects to be included upon the second GenerateAC (CDOL2).
– The list of objects to be included upon internal authentication (DDOL):
– Tag `0x9F37` – "Unpredictable Number" (length: 4 octets)
– Cardholder's name: abridged here as H.D. for privacy reasons.

The chip belongs to Mr. H.D., which is also inconsistent with the information embossed on the card.

ReadRecord SFI 3 record #1, 2, 3, 4 (used for "disconnected" mode data authentication) contained actions codes, requested by the card to authorize a transaction, as well as a list of supported authentication methods, their public keys and certificates.

ReadRecord SFI 1 record #1 should have revealed the exact same information encoded in the ISO2 track. Instead, it contained, again, the following information:

– Account number: `4561**********79`
– Expiration date (YYMM): a date in 2011 (anonymised for privacy reasons)

ReadRecord SFI 4 record #1 indicated an empty record.

3.3 Authentications

InternalAuthenticate In smart card terms, an InternalAuthenticate[19] is an authentication of the card by the reader (*cf.* Chapter 6.5 of [4]). The reader requests that the card signs a random 4-byte number, as asked by the DDOL.

The reader accepted this authentication.

VerifyPIN (Cardholder verification) The reader checks that the card isn't blocked by reading the number of remaining PIN presentation tries[20]. There are 3 remaining tries before the card is blocked.

PIN codes are verified using the command VerifyPIN[21]. A correct PIN results in a `0x9000` status word.

Our experiments reveal that the PIN is always considered correct, regardless of P1 and P2, even for inconsistent values. The card accepts any PIN unconditionally.

[19] Command: `00 88 00 00 04`.
[20] Command: `80 CA 9F 17 04`.
[21] Command: `00 20 00 80 08`.

3.4 Transaction

The reader gathers risk management data before starting a transaction.

GetData (ATC) The ATC (Application Transaction Counter) was requested[22].

The ATC sent by the card does not change, regardless of the number of transactions performed. This ATC is different from the one returned by the first GenerateAC (which is incremented at each transaction), and is therefore clearly false.

The ATC is forged to manipulate the PoS risk management routine, which would otherwise request to go on-line.

The above also applied to the reading of the last online ATC[23].

Risk management Based on available data, the reader performs risk management as described in Chapter 10.6.3 of [5]:

> "*If the required data objects are available, the terminal shall compare the difference between the ATC and the Last Online ATC Register with the Lower Consecutive Offline Limit to see if the limit has been exceeded.*"

Here, ATC (`0x04`) – LOATC (`0x02`) > LCOL (`0x01`).

As the transaction log extracted from the card indicated, the fraudsters performed many small amount purchases to avoid on-line connection requests.

4 Side-channel Power Analysis

Measuring variations in the device's power consumption enables detecting patterns that correspond to repeated operations. This is a common way to try and determine secret keys used in cryptographic operations. Although very rarely, side-channel analysis is also used by forensic experts (*e.g.* [9]).

Here, side-channel analysis will expose the fact that the forgery contains an underlying (legitimate) card, by analysing in detail the forgery's power trace when it is operated.

We shall constrast the "VerifyPIN" command, which does not propagate to the stolen card, with the "Select" command, which must be relayed to the stolen card.

4.1 EMV "Select" command

The VISA application is selected based on its AID.

The sequence diagram of Figure 10 shows what should happen if the forgery indeed behaved as a "chip-in-the-middle" proxy.

Power consumption is measured and synchronized with the I/O between the card and the reader. However, internal communication between the FUN card and the stolen chip is witnessed on the power trace.

Figure 11 shows power consumption over time when the Select command is sent to the forgery. Patterns clearly appear during I/O activity. Some patterns can also be noticed *between* I/O operations, while there is no communication with the reader. A finer analysis shows that these patterns are made of sub-patterns, whose number is equal to the number of bytes exhanged. This confirms that communication is intercepted and retransmitted by the FUN card, as illustrated in Figure 12.

[22] Command: `80 CA 9F 36 05`.
[23] Command: `80 CA 9F 13 05`.

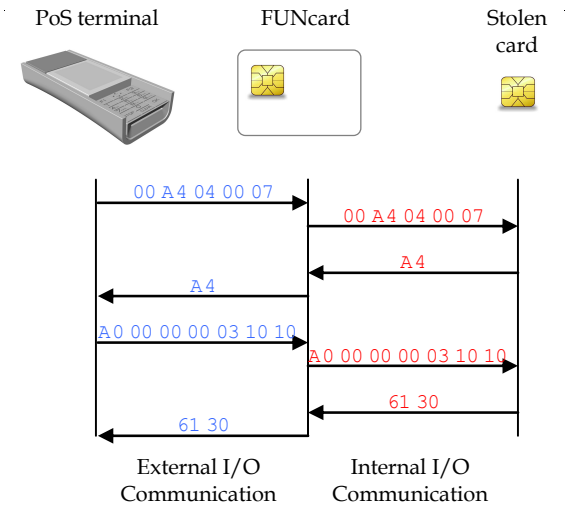

Fig. 10. The FUN card intercepts the Select command and replays it to the stolen card. Then the FUN card sends back the response to the PoS.

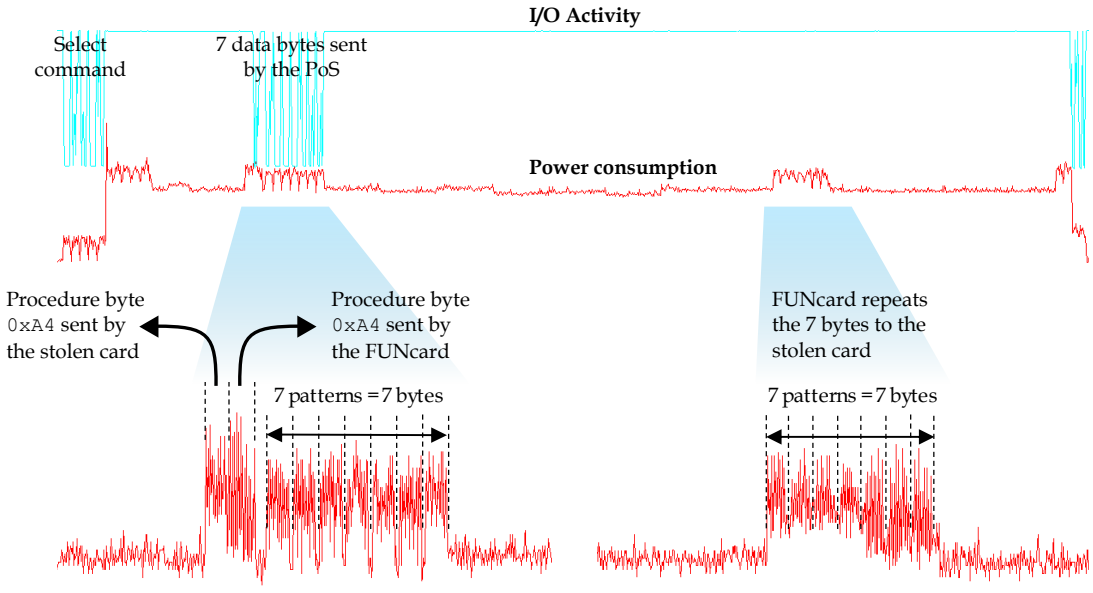

Fig. 11. The power trace analysis of the forgery during the Select command reveals a pattern that is repeated, despite the absence of I/O operations. It is readily observed that the pattern corresponds to the replay of data sent earlier by the PoS.

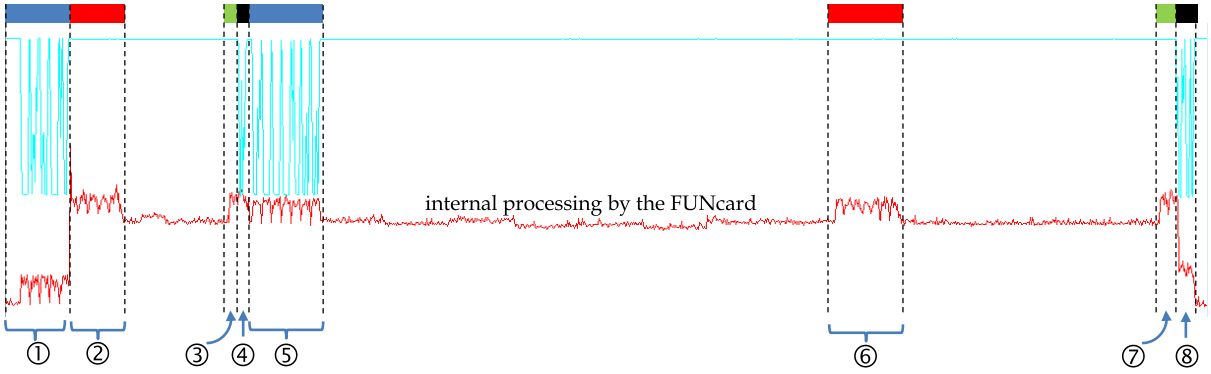

Fig. 12. (1) PoS sends the ISO command `00 A4 04 00 07`; (2) The command is echoed to the stolen card by the FUN card; (3) The stolen card sends the procedure byte `A4` to the FUN card; (4) The FUN card retransmits the procedure byte (`A4`) to the PoS; (5) The PoS sends the data `A0 00 00 00 03 10 10` to the FUN card; (6) The FUN card echoes `A0 00 00 00 03 10 10` to the stolen card; (7) The stolen card sends the status word (`SW1=61`, `SW2=30`) to the FUN card; (8) and the FUN card transmits `SW1 SW2` to the PoS. Communication: PoS → FUN card is shown in blue; FUN card → stolen card in red; Stolen card → FUN card in green and FUN card → PoS in black.

4.2 EMV "VerifyPIN" command

We now turn our attention to the "VerifyPIN" command which, in the case of a proxy chip, would never be sent to the stolen chip.

As expected, this command is executed directly, as shown on the power trace of Figure 13. No operations between I/Os are witnessed here.

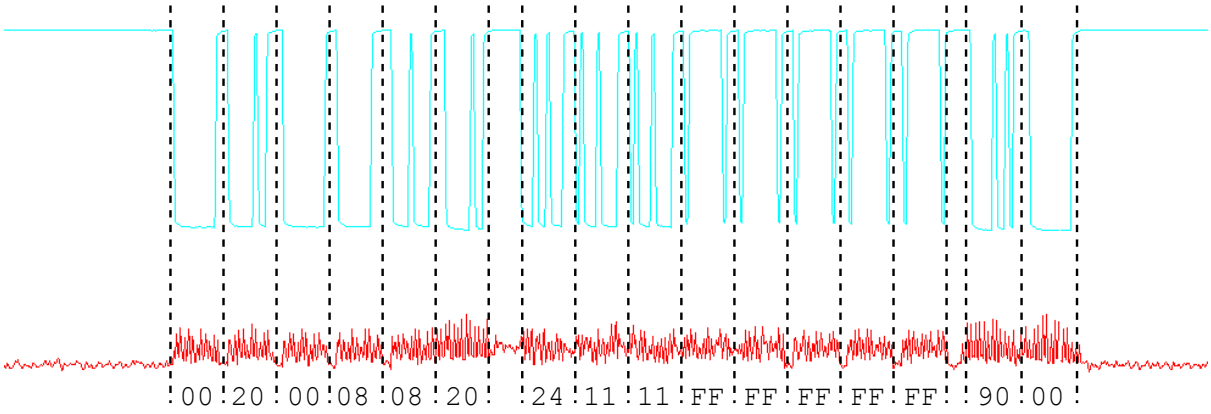

Fig. 13. Power trace of the forgery during the VerifyPIN command. Notice the absence of a retransmission on the power trace before the returning of `SW1 SW2`.

4.3 GetData commands

When sent a GetData command, the card seems to modify some values used for risk management purposes, so as to manipulate the PoS. The level of resolution offered by power trace analysis (Figure 14) is insufficient for seeing when this happens.

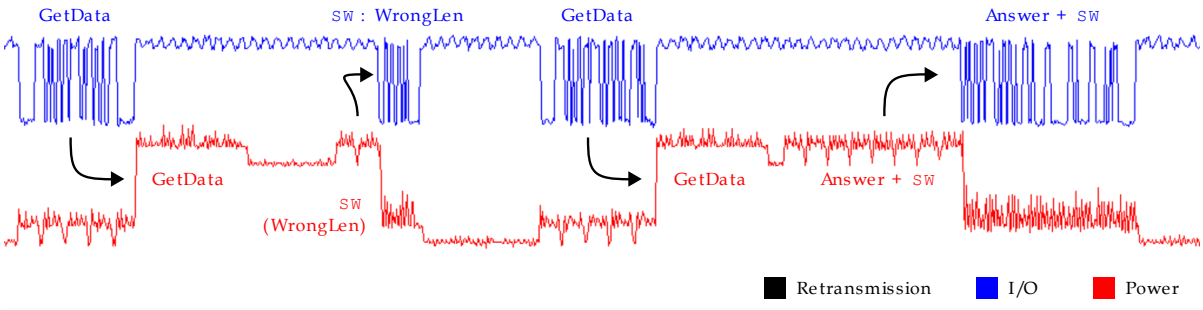

Fig. 14. Power consumption during a GetData command.

5 Destructive Analysis

We finally de-capsulated the forged module to analyze its internal structure. Results are shown in Figures 15, 16, and 17.

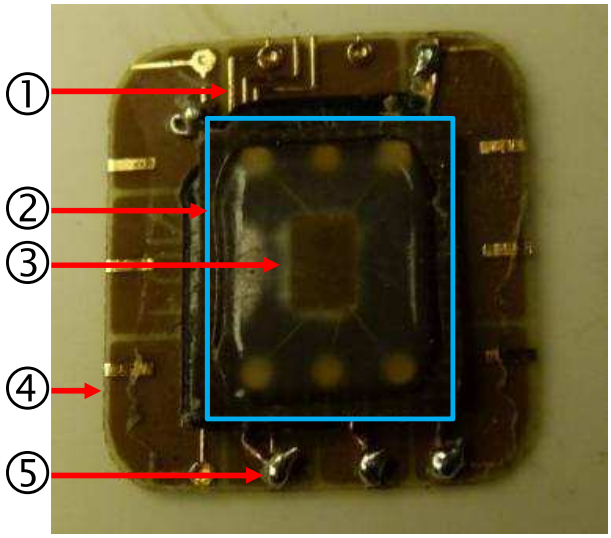

Fig. 15. (1) Connection grid; (2) Stolen card's module (outlined in blue); (3) Stolen card's chip; (4) FUN card module; (5) Weldings of connection wires.

The `Vcc`, `RST`, `CLK`, `GND` contacts of the FUN card are connected to the corresponding pins of the stolen card (`Vcc` to `Vcc`, `RST` to `RST` etc.). However the stolen card's `IO` pin is connected to the `SCK` pin of the FUN card (Figure 18).

5.1 Anti-Forensic Countermeasures

Figure 19 shows that the perpetrators scratched the printed circuit copper track of `SCK` to conceal the traffic transiting *via* `SCK`. Recall that `SCK` is the most informative signal in the device because `SCK` is used by the FUN card to communicate with the stolen card.

During questioning by law enforcement, two reasons were advanced by the perpetrator for doing so. The first was, indeed, the intention to make forensic analysis harder. The second is way more subtle: using a software update, PoSs could be modified to spy the traffic on `SCK`. This would have allowed deploying a software countermeasure that would have easily detected forged cards.

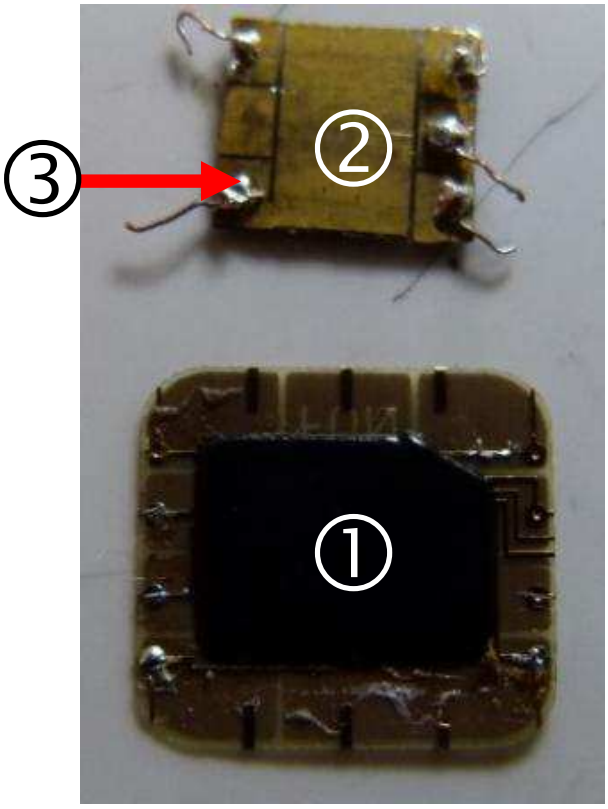

Fig. 16. (1) FUN card module; (2) genuine stolen card; (3) welded wire.

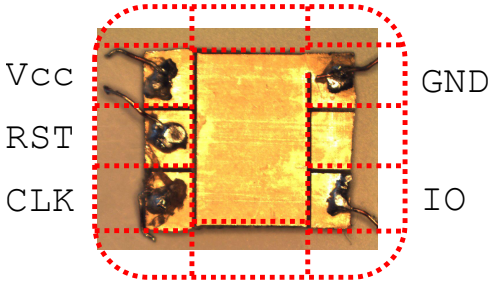

Fig. 17. Original EMV chip clipped by the fraudsters, with the cut-out pattern overlaid.

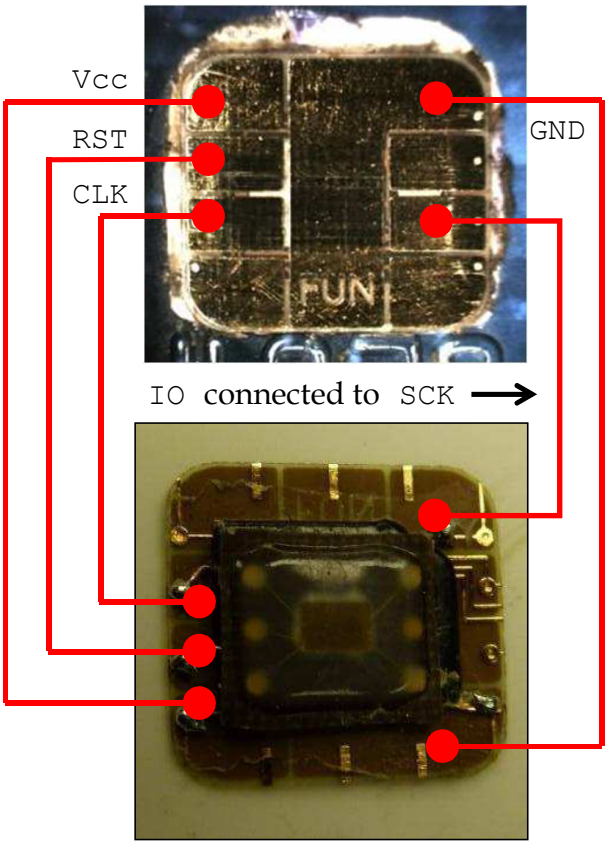

Fig. 18. Wiring diagram of the forgery.

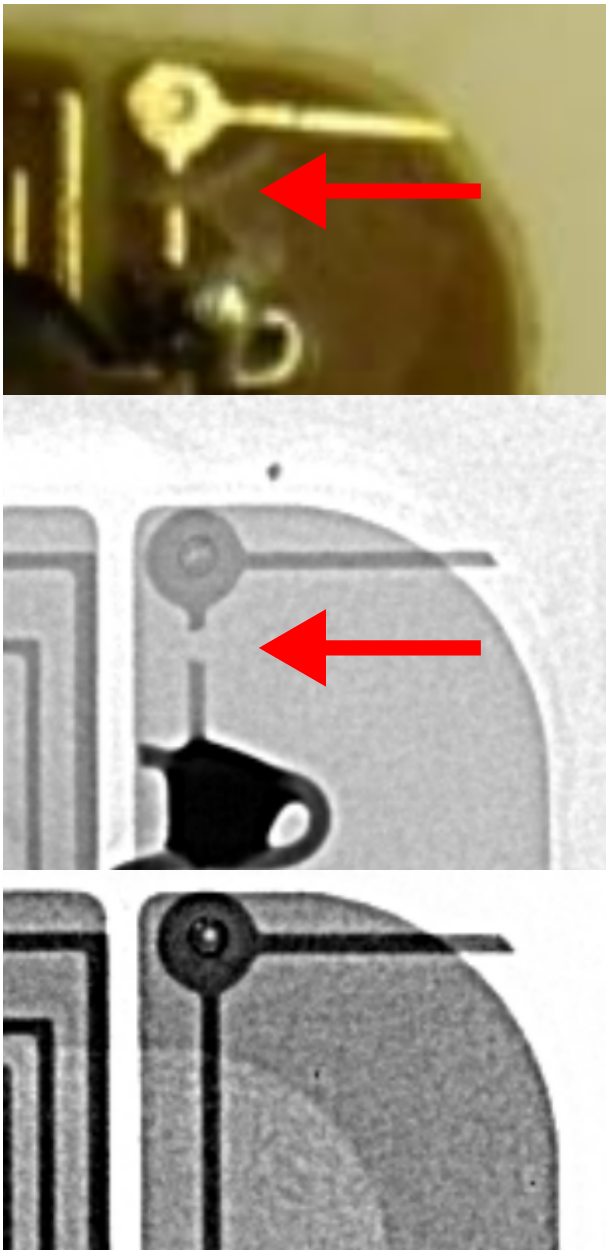

Fig. 19. Anti-forensics precautions taken by the perpetrator. Zoom on parts of Figures 18 (fraudulent card), 6 (X-ray of the fraudulent card), and 7 (unmodified FUN card). The abrasion and cut wire are clearly visible.

6 Aftermath & Lessons Learned

The forensic report produced by the authors of this paper was sufficient for the court to condemn the perpetrators. During our testimony we underlined to the court that this case shows that organised crime is following very attentively advances in information security. We also noted that producing the forgery required patience, skill and craftsmanship. It is important to underline that, as we write these lines, the attack described in this paper is not applicable anymore, thanks to the activation of a new authentication mode (CDA, Combined Data Authentication) and network level protections acting as a second line of defense. Until the deployment of CDA, this fraud was stopped using network-level counter-measures and PoS software updates. While we cannot detail the network-level countermeasures for confidentiality reasons[24], the following two fixes allowed to immediately stop the fraud:

Parity Faults We assumed that the fraudster only implemented the just-enough functionalities allowing to perform the fraud. This was indeed the case: when injecting byte-level parity errors into bytes transmitted from the PoS to the FUN card, the FUN card did not request byte retransmission as mandated by the ISO/IEC 7816 standard. Coding, testing and deploying this countermeasure took less than a week.

Abnormal Applicative Behavior The forged card replies with a status word equal to `0x9000` to VerifyPIN commands sent *outside* of a transaction context (*e.g.* just after a reset). This is incompliant with EMV specifications (where a PIN is necessarily attached to a previously selected application) and proves that the FUN card is not context-aware. Coding, testing and deploying this countermeasure was done overnight.

In addition, four other software-updatable countermeasures were developed and tested, but never deployed. These were left for future fraud control, if necessary.

Nonetheless, this case illustrates that, as a rule of thumb, an unmalleable cryptographic secure channel must always exist between cards and readers. Other (more expensive) solutions allowing to avoid man-in-the-middle devices consist in relying on physical attestation techniques such as [6].

7 Other Applications of Miniature Spy Chips

The technique explained in this paper can be generalized to attack non-payment devices.

7.1 Eavesdropping Mobile Communications

By extracting a chip from a FUN card and implanting it under the SIM connector of a smartphone, mobile communications can be monitored and decrypted. The demonstrator, on which we currently work, functions as follows: GSM and 3G communication confidentiality is based on session keys (denoted Kc, CK and IK) transmitted by the SIM to the phone. These session keys are derived from a random challenge (RAND) sent from the Authentication server (AuC) to the SIM (see Figure 20). A FUN card implanted under the reader can easily monitor these exchanges and record Kc, CK and IK in EEPROM.

While this happens, the opponent intercepts and records encrypted voice communications without decrypting them. It remains to extract the captured key material and transmit it to the attacker. This is far from being a trivial task given that, unlike the EMV fraud case that we have just analyzed, a FUN card implanted under a card reader does not actively control the SIM.

[24] These can potentially be efficient against yet unknown future forms of fraud.

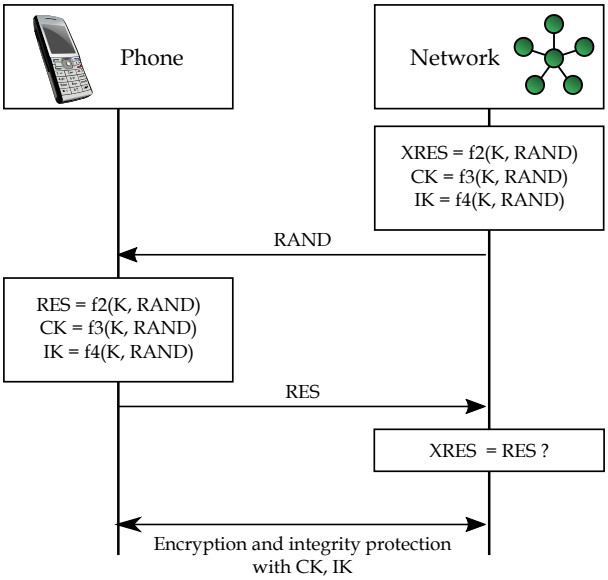

Fig. 20. 3G authentication protocol (simplified).

As strange as this may sound, as a matter of fact *it does*, assuming that the FUN card can read bits quicker than the phone (which is the case in practice). The ISO/IEC 7816 protocol relies on the fact that the `IO` signal connecting the card to the reader is pulled-up. This means that a party wishing to communicate pulls-down the `IO` and hence signals a zero to the other party. When the communicator's port is switched to high impedance, the line automatically goes up again. Hence, if we connect the FUN card's `IO` to the SIM connector's `IO`, both the FUN card and the legitimate SIM can signal zeros to the phone. In other words, the phone will see the information $b_f \wedge b_s$ where b_f and b_s (respectively) denote the bits sent by the FUN card and by the SIM.

To prove its identity to the network, the SIM returns to the AuC a response called SRES (or RES). Hence, the FUN card can intervene in the transmission of RES and force some of RES's bits to zero. Because RES is false the authentication will fail *but* information (in which the FUN card can embed Kc, CK or IK) will be broadcast to the attacker over the air. This precise information encoding problem was already considered by Rivest and Shamir in [8].

The implementation of this strategy is technical. It requires more than just turning bits to zero, because every byte sent from the SIM to the phone has a parity bit. Switching a single bit to zero means that the parity bit must also be flipped, which can only be done when the parity is one. Hence, the FUN card needs to compute the parity p of bits [0:6]. If $p = 0$ or bit 7 is zero, the FUN card remains quiet. Else, the FUN card pulls down the `IO` during bit 7 *and* during the parity. Another option consists in pulling down two data bits during transmission and leaving the parity unchanged.

7.2 Characterising Unknown Readers

Consider the case of a border control device, produced and sold in small quantities, to carefully chosen clients. Users are given identification cards that interact with the device, but the description of the ISO commands exchanged between the card and the device is kept confidential. Exhausting all possible commands is impossible because critical-infrastructure cards usually embed a ratification counter that limits the number of unknown commands to 10 before definitively blocking the card.

An intelligence agency wishing to characterise the readers and understand how they work may construct a "chip-in-the-middle" command recorder based on a FUN card and a genuine identification card. The ISO command set could then be retrieved for later analysis.

7.3 Low-Cost Hardware Security Modules

Fig. 21. An industrial multi-SIM reader.

In a number of industrial settings, keys, signatures or ciphertexts must be generated at a fast pace. A smart-card has relatively strong tamper resistance defences but modest computational capabilities. Hardware Security Modules (HSMs) are expensive devices featuring both tamper-resistance and important computational capabilities.

A number of manufacturers propose multi-smart-card readers (Figure 21). In such readers, a central processor establishes distinct one-to-one connections with each smart-card. An alternative HSM concept, illustrated in Figure 22, would consist in simply wiring card modules to each other. Power and clock supply would be common to all card modules. All card modules will be reset at once (given that `IO` is pulled up, the simultaneous emission of answers to reset will not cause signal conflicts). Communicating with individual modules will only require the (software) coding of a non-ISO protocol, where modules monitor `IO` and emit information to the reader while avoiding collisions.

References

1. French prosecution case number 1116791060.
2. EMVCo: `http://www.emvco.com/specifications.aspx`
3. EMVCo: EMV Specification (Book 1) – version 4.2 (June 2008)
4. EMVCo: EMV Specification (Book 2) – version 4.2 (June 2008)
5. EMVCo: EMV Specification (Book 3) – version 4.2 (June 2008)
6. Mayes, K., Markantonakis, K., Chen, C.: Smart card platform fingerprinting. The Global Journal of Advanced Card Technology pp. 78–82 (2006)
7. Murdoch, S.J., Drimer, S., Anderson, R., Bond, M.: Chip and pin is broken. In: 2010 IEEE Symposium on Security and Privacy. pp. 433–446. IEEE (2010)
8. Rivest, R.L., Shamir, A.: How to reuse a "write-once" memory. Information and control 55(1), 1–19 (1982)
9. Souvignet, T., Frinken, J.: Differential power analysis as a digital forensic tool. Forensic science international 230(1), 127–136 (2013)

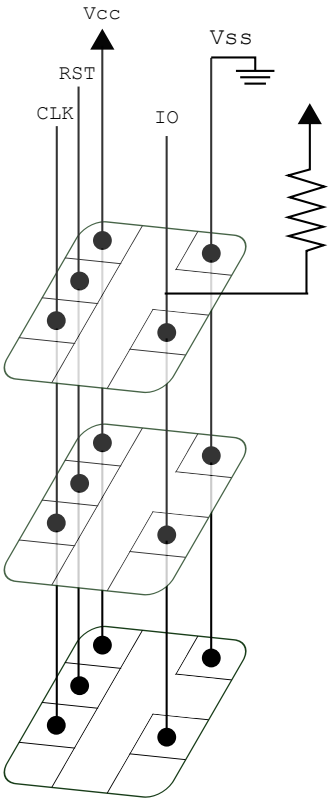

Fig. 22. Proposed low-cost HSM design based on SIM cards.